Fuzzy Cellular Automata
in
Conjunctive Normal Form

by David Michael Forrester

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfilment of the requirements for
Master of Computer Science
May 11, 2011

uOttawa

Ottawa-Carleton Institute for Computer Science
Faculty of Graduate and Postdoctoral Studies
School of Information Technology and Engineering
University of Ottawa

Abstract

Cellular automata (CA) are discrete dynamical systems comprised of a lattice of finite-state cells. At each time step, each cell updates its state as a function of the previous state of itself and its neighbours.

Fuzzy cellular automata (FCA) are a real-valued extension of Boolean cellular automata which *"fuzzifies"* Boolean logic in the transition function using real values between zero and one (inclusive). To date, FCA have only been studied in disjunctive normal form (DNF).

In this thesis, we study FCA in conjunctive normal form (CNF). We classify FCA in CNF both analytically and empirically. We compare these classes to their DNF counterparts. We prove that certain FCA exhibit chaos in CNF, in contrast to the periodic behaviours of DNF FCA. We also briefly explore five different forms of fuzzy logic, and suggest further study. In support of this research, we introduce novel methods of simulating and visualizing FCA.

Acknowledgements

I would like to thank my supervisor, Paola Flocchini, for her guidance and insight. I would also like to thank my parents, Michael and Lisette Forrester for their continued support.

I dedicate this work to my beloved pet cat Felonious, who died during the writing of this thesis.

Contents

Chapter 1

Introduction

1.1 The Inception of Cellular Automata

It has long been the ambition of scientists to better understand the enormous complexity of the universe in which we reside. Biological systems are some of the most complex systems ever discovered, and can exhibit traits not found elsewhere in nature. In the biological world, characteristics such as self-organization, self-replication, and chaos are not uncommon. The DNA found in living organisms behaves as a self-propagating information storage and processing machine. The central nervous system found in higher organisms also behaves as a massively parallel, self-sustaining computer. Understanding how these systems function is challenging enough; replicating these behaviours has proven to be extremely difficult.

That is exactly what esteemed mathematician John von Neumann set out to do in the late 1940s with his work on self-replicating machines [53]. While the creation of self-replicating mechanical machines was technologically infeasible at the time, von Neumann attempted to create simplified mathematical models that could demonstrate

abstractly how self-replicating machines might work. He based his model on the level of abstraction of the living cell.

The problem was then naturally divided into two parts: the first part required an understanding of how the cells themselves would work; the second part involved understanding how these cells could be arranged into patterns, or *organisms*, that could perform more complex tasks. Since von Neumann was interested in simplifying his models, he set aside the problem of understanding how individual cells functioned, and instead considered cells as automatisms, or "black boxes", which had unambiguous defined responses to external stimuli. He then set about deciding how cells would respond to their environment, and how they could be arranged to perform complex tasks such as computation and self-replication.

His first, and most general, model was termed the *kinematic model*, where he envisioned a set of robotic elements such as sensors, tools, and structural elements, as well as signal processing and memory elements. These ideas were quite general and abstract, and implementation was infeasible, but this formed the basis for his subsequent work.

Von Neumann decided to study a simplified model that did not include any of the mechanical aspects of the kinematic model, instead focusing solely on computation. He called this the *cellular model*. In doing so, he hoped to demonstrate trivial cases of what he saw in the natural world. The ultimate goal of his research was to try to reconcile the vastly divergent fields of information science and biology.

Still, von Neumann struggled with the complexity of his models. His colleague, Stanisław Ulam, had been studying crystal growth, and suggested that his simplified physical models of crystals could be used to simplify von Neumann's models of cellular machines. Von Neumann made the following restrictions to his model, and in doing

so, created the first example of what is now known as a *cellular automaton*.

First, he restricted the physical space to a discrete set of points, arranged on a two-dimensional plane in a regular grid, or mesh. Each point in this mesh was called a cell, and each cell could assume any one of a discrete and finite set of states. Each cell would act as its own finite state machine, moving from its current state to a new state, using the states of itself and surrounding cells as input. Each cell is identical, in the sense that each cell behaves the same as any other when given identical input. Each cell is anonymous, in the sense that a cell is not aware of its position or role in the larger scheme of things. Finally, the temporal space of this dynamical system was made discrete, so that updates occurred instantaneously, and simultaneously for all cells in discrete time steps. While variations are possible, this is the basic definition of what is now known as a *cellular automaton* (plural: *cellular automata*, abbreviated as a *CA*).

Von Neumann was interested in the biological process of self-replication. To model this, his first CA consisted of cells with 29 possible states that were able to transform a nearby area of the grid into a copy of itself. To achieve this, he solved an even greater problem: creating a *universal constructor* (fig. 1.1).

Figure 1.1: Pesavento implementation of the Von Neumann Universal Constructor [47]

The universal constructor is not only able to create a copy of itself, but it can

transform the surrounding region into any conceivable configuration by reading instructions stored as a string of cells, known as a *tape*. In the case of replication, the tape not only contains instructions to construct a copy of the constructor, but to copy the instruction tape itself. This could conceivably allow the constructor to create an infinite number of copies of itself. Von Neumann touted his system as a simplified model of biological asexual reproduction, where the input tape was analogous to DNA, and the constructor itself being analogous to a living organism.

Von Neumann's models have since been shown to be inadequate at capturing the properties of real living cells. However, this research has spawned an entire field of research into these abstract mathematical objects.

1.2 Significance and Properties of Cellular Automata

Von Neumann's results contained a key property of CA that went largely ignored until many years later. First, not only were cellular automata capable of computation, but cellular automata are capable of *universal computation* (also known as being *Turing complete*). That is, it is possible to create a CA that can perform any possible classical logical computation. In short, a CA can act as a computer.

A computationally universal CA has several important properties: its processing is massively parallel, and homogeneous, consisting only of local interactions. These properties have two important implications: a CA could conceivably be used to simulate the physical world, and the physical world could conceivably be used to construct a CA.

CAs have been used to model and simulate fluid dynamics, gas diffusion, wave propagation, surface tension, wetting, packing, crystal growth, vehicular traffic, ant

colonies, etc. [14, 54, 3, 33, 18, 38]. Examination of the source code of the popular computer game Sim City from 1989 reveals that it, too, was modelled on a CA [58]. It has even been argued by both philosophers and physicists that the universe itself *is* a cellular automaton [48, 61]. But the very properties that make cellular automata attractive for modelling and simulation (massively parallel, and localized), make them difficult to simulate on a traditional computer which processes information linearly. Because of this, CA were largely ignored as a suitable method for modelling and simulation until very recently. And as a consequence of the halting problem, there is no way of determining a future state of a universal CA without actually performing a complete simulation [51]. Physicist and mathematician Stephen Wolfram, a fervent proponent of CA modelling, called this *computational irreducibility* [57]. He argues, sometimes controversially, that nature itself exhibits simple localized computations executed in parallel that are irreducible, and that in some cases, CA could model the complexity found in nature better than traditional mathematical models.

Inversely, it is conceivable that a physical implementation of a cellular automaton could act as an immensely powerful computer. In theory, a CA computer whose cells exist at the molecular, atomic, or subatomic scale could approach the theoretical limits of information storage and processing density. Several computer architectures have been proposed that are based on cellular automata [24, 40], including some that extend CA into the realm of quantum computing [35]. A CA architecture has several desirable qualities. Again, such a computer could perform massively parallel computations easily. The homogeneous nature of CA would eliminate the distinction between the computer's memory and processor, simplifying design. Since information is passed directly between adjacent cells, it would eliminate the need for data transmission buses and wires. Also, since CA cells are autonomous, a CA architecture would be fault tolerant; one part of the processor could continue to function even if another

part has been destroyed. While a true CA computer has yet to be built, current state-of-the-art multi-core parallel computer architectures, which are not a CA in the strictest sense of the word, do share many of its properties.

While cellular automata have many applications, our focus will be on their underlying mathematical behaviour.

Since von Neumann's original work, much work has been done to define the properties of cellular automata. Von Neumann made many arbitrary choices in the construction of his automata, such as the number of possible states of each cell, and the choice of lattice on which cells existed. Would properties such as universal computation hold if the complexity of his models were reduced? In 1983, Stephen Wolfram introduced the least complex mathematical system that could still be considered a CA, called an *elementary cellular automata* (see Section 2.2) [55]. In doing so, he provided the simplest possible system for mathematicians to work with in their attempts to define the nature of CA.

Classifying cellular automata has proven to be an important first step in their study. By studying CA rules in groups rather than individually, conclusions can sometimes be made about the properties of an entire class, which in turn may help explain the observed behaviour of that class. Shortly after their introduction, Wolfram attempted to classify elementary CA empirically, based on their asymptotic behaviour [56].

He observed that some of the elementary CA in one of his empirical classifications exhibited complex "chaotic-like" behaviour. A chaotic system is a dynamical system that is highly sensitive to initial conditions [37, 2]. This means that a chaotic system will become widely divergent with time, despite being completely deterministic in nature. While there are many different definitions of chaos, we refer to the one formally

defined in [4].

Certain elementary CA rules have been found exhibit chaos in the classical sense [9, 11, 12]. However, the metric typically used in dynamical systems defines chaos for very simple observable behaviors: for example, a simple shift would be chaotic according to the classical definition. On the other hand, chaos would be expected to occur when the dynamics is much more complex. This raises an important question on the meaning of chaos in *discrete dynamical systems*, which has not been answered yet.

Other classifications of ECA have been proposed in the literature (e.g., see [1, 8, 22, 36, 49]), each focusing on different grouping criteria. In all cases, the most interesting observable behaviors correspond to the formation of random-like complex patterns. Although these patterns have never proven to be associated with true chaos, their dynamics has been extensively investigated (e.g., see [13, 26, 34, 46, 50]).

1.3 Extensions of Boolean Cellular Automata

A class of objects similar to cellular automata known as *coupled map lattices* (CML) were introduced by K. Kaneko in 1984 as a means of studying physical systems. [27]. A coupled map lattice is similar to a CA, but differs from a CA in that its state space is continuous, rather than discrete, and the local transition function is in the domain of real numbers. Coupled map lattices are a very general class of objects, and may exhibit a range of complex behaviours [28, 29, 30].

Fuzzy cellular automata (FCA) were introduced in [10] as a particular type of CML that is a continuous-valued version of elementary discrete CA. FCA employs the concept of *fuzzy logic* introduced by Lotfi A. Zadeh in 1965. [59]. This “many-valued

logic" extended the notion of *true* and *false* to include in-between values. Taken to the extreme, fuzzy logic can include an infinite number of states between true and false, known as continuous fuzzy logic. That is, rather than having cells which could assume only binary values, FCA cells could assume any value between zero and one, inclusively. Boolean operators are also "fuzzified" by replacing them with standard algebriac operators. FCA use this process to "fuzzify" the Boolean logic of a corresponding Boolean CA that is expressed in some normal form. Generally, FCA have been studied in disjunctive normal form (DNF) (see section 2.2.3). Certain equivalences were shown between regular elementary CA their elementary FCA counterparts in DNF (see Section 4.2) [6]. It was also shown that none of the elementary FCA exhibited chaos, and all had a periodic asymptotic behaviour [5, 6, 41, 42].

1.4 Objectives and Motivation

To date, elementary fuzzy cellular automata have only been studied in the disjunctive normal form (DNF) (see section 2.2.3), and with a particular type of fuzzy logic known as CFMS (see section 2.3). This combination has been studied extensively, as it was discovered that if the value of a cell was interpreted as the probability of that cell being one in the equivalent Boolean CA, then the application of the local fuzzy transition function gives the probability that that the cell will be one in the next generation [19].

All subsequent work on FCA has been bound to this choice of canonical form and type of fuzzy logic. In DNF with CFMS, FCA have been classified analytically, and a correlation has been shown to exist between FCA in DNF, and Boolean CA. This has made FCA in DNF with CFMS a useful tool in understanding the behaviour

of corresponding Boolean CA.

Our main objective is to begin studying FCA in conjunctive normal form (CNF) (see section 2.2.3). In particular, we wish to classify FCA in CNF in a manner similar to what has been done in DNF. This means that we will classify CNF analytically when possible, and empirically when mathematical analysis proves infeasible. Empirical classification will entail the simulation and visualization of FCA in CNF. We will then investigate the properties of these new classes, should any of them prove interesting.

Our secondary objective relates to the choice of *fuzzy logic* and table 2.4). but other logics exist. We will perform a cursory analysis of FCA in other logics to determine if further study is warranted.

1.5 Results of the Thesis

The results of our thesis can be divided into three main topics:

Simulation and Visualization: Empirical classification of complex abstract mathematical objects necessitates computer simulation. As a prerequisite to performing these simulations, we have written a software application to simplify the creation and visualization of cellular automata. We introduce a new and flexible method of defining a CA's local transition function as a computer program that can be compiled and executed on-the-fly. With the aid of this software, we also explore visualization options, such as mapping values to colour gradients, and using colour quantization to better visualize the mathematical structure of cellular automata.

Classification of Fuzzy CA in Conjunctive Normal Form: We explore elementary FCA in CNF, and make the following discoveries: We classify homogeneous FCA in CNF analytically. We prove that certain FCA exhibit chaos in CNF, in striking contrast to the periodic behaviours of DNF FCA. We classify random FCA in CNF empirically, by simulating them with our CellView software.

Classification of Fuzzy CA with Different Logics: We consider CNF fuzzification with 5 other logics. We derive their analytical form and we observe them to determine if they are of interest for further study. In particular, we analyze homogeneous elementary FCA. In each case, we are able to show the asymptotic behaviour of these CA analytically by solving them algebraically. We find that elementary CA in these other logics have extremely simple behaviours that do not capture many of the properties of their Boolean counterparts.

1.6 Breakdown of Thesis Structure

In **Chapter 2**, we define several key concepts that may be required in understanding the rest of the thesis.

In **Chapter 3**, we introduce the CellView software used to conduct some of the experiments that appear later in the thesis. We discuss simulation and visualization features that are important to CA research in general.

In **Chapter 4**, we provide background of some previous research that has been done with FCA in DNF. We define how FCA in DNF have been classified, and discuss the relationship between FCA and Boolean CA. We mention that FCA in DNF do not exhibit chaos. Finally, we demonstrate some applications of FCA.

Chapter 5, comprises the main results of our thesis. We introduce FCA in CNF, and introduce a naming convention used to identify them. We classify FCA in CNF analytically, in the special case of a homogeneous configuration. We discover that one such class exhibits chaos, in the classical sense. We then classify FCA in CNF empirically, in the general case.

In **Chapter 6**, we perform a cursory analysis of FCA in CNF with other forms of fuzzy logic. We show that in the homogeneous case, they exhibit simple behaviour.

Chapter 7, we discuss our results, ask open questions, and suggest further research.

Chapter 2

Definitions and Notations

In this chapter, we introduce several definitions that will be used later in the thesis.

2.1 Cellular Automata

2.1.1 Fundamentals

A *cellular automaton* (CA) is a dynamical system which is composed of a regular lattice of anonymous, homogeneous *cells* that change with time.

Cells are arranged in some form of *regular lattice*. The most common lattice by far is the d-dimensional rectangular mesh, denoted $\mathbb{Z}^d$. We will restrict our discussion to the 1-dimensional linear array.

At any point in time, each cell has a particular state. The set of all possible states for any cell is defined as the finite state set S. In a cellular automaton, every cell shares the same state set.

The state space of the entire automaton is the union of the state sets of all cells, $S^{\mathbb{Z}^d}$, denoted as $C(d, S)$, or simply C [32]. This is called the *configuration space*. Each element of the configuration space represents one possible configuration of the automata as a whole, which is a mapping $X : \mathbb{Z}^d \to S$ that specifies the current states of all the cells.

The region of cells that a cell can observe is called its *neighbourhood*. Each cell's neighbourhood is defined as a region relative to its own location. A neighbourhood definition is created as list of adjacent cells using local relative addressing. The neighbourhood vector, N, is defined as $N = (x_1, \ldots, x_m) \in \mathbb{Z}^m$, where m is the number of cells in the neighbourhood, called its size.

Cellular automata have a temporal space that is *discrete* and *synchronous*. Time is discrete in that time advances in discrete steps, and the automaton instantaneously changes from one state to another state with each time step. Time is synchronous in that time advances simultaneously for all cells in the automaton.

At each time step, each cell updates its state to a new state based on the *local transition function*, denoted as $f : S^m \to S$. The local transition function takes the current state of the cell and its neighbours as input, and produces the next state as output. Each cell executes the local transition function independently. Although different cells may have different states, they all share the same state space, and they all share the same local transition function. For this reason, we say that cells are *homogeneous*.

The automaton's *configuration*, denoted as X, is an aggregation of the state of all cells. This is represented as a vector. The configuration can also be represented as a mapping that maps each cell's address on the lattice to a state: $X : \mathbb{Z}^d \to S$. At time $t = 0$, the configuration is not derived, but defined. We call this the *initial*

configuration.

The set of all possible configurations is called the *configuration space*, and is denoted as C. This represents all possible global states that the CA can assume.

The global transition function, denoted as F, represents the application of the local transition function f on all cells. It is a mapping from one configuration to another: $F : C \to C$. The global transition function defines the evolution of the CA as a whole.

So, a cellular automaton $\mathcal{A}$ can be defined as a 3-tuple system: $\mathcal{A}$ = (S, N, f).

2.1.2 Boundary Conditions

As we have seen, a CA consists of cells arranged in some regular lattice, or mesh. Care must be taken to define the size of this mesh. A CA may be of either *finite* or *infinite* size.

A finite sized CA, by definition, must have some boundary that defines the edge of its topology. This presents a contradiction: all cells are homogeneous, however cells lying along the edge of the boundary have fewer neighbours than interior cells. Since all cells must have the same neighbourhood, one must define how boundary cells react to this condition.

One option is to permanently define the value of any missing neighbours to be the *quiescent state*, which is usually zero. One can imagine this as a dynamic, finite CA existing inside an infinite, static CA. We call this having a *quiescent background*.

A second option is to define the address of each cell in n-modulo arithmetic for each dimension, where n is the size of the dimension. We say that the CA is *circular*

(or *toroidal* in the 2-dimensional case), as the edge of the CA "wraps around" to the other side.

A CA of infinite size in all dimensions and directions has no boundary. We say it is it has an *infinite background*. It is also possible for a CA to be one-sided infinite, or to be infinite in only a subset of its dimensions. In each of these cases, such a CA has an infinite number of cells. Simulating an infinite CA is impossible, except in certain exceptional cases described below.

A one-dimensional infinite CA may be simulated as a finite, circular CA of size n in the case where the configuration takes on the repeating form: $X = (x_0, x_1, \ldots, x_{n-1})^\infty$. We say that such a configuration is *spatially periodic*. In the extreme case, a CA can have the form $X = (x)^\infty$. This is called a *homogeneous configuration*.

It is also possible to simulate a finite subset of an infinite CA, to a finite number of time steps. To simulate a subset of length n of an infinite one-dimensional CA with a neighbourhood of radius 1 to t time steps, simply simulate the subset as a finite, circular CA with t extra cells added to each boundary. So, if the subset is of length n, the total size that needs to be simulated is $n + 2t$.

Consider now *circular* CA. As mentioned, a circular CA differs from a bi-infinite CA in that there are a finite number of cells, and the last cell is considered adjacent to the first. The relative indexing used to calculate neighbours then uses *modulo* n arithmetic, where n is the length of the CA. With a circular CA, the initial configuration can be expressed as $X^0 = (x_0^0, x_1^0, \ldots, x_{n-1}^0)$, and the configuration at time T as X^t.

A CA is said to be *temporally periodic* with period τ if:

$$\exists T \mid \forall t > T : F(X^t) = F(X^{t+\tau})$$

A CA is said to be *spatially periodic* with period ω if

$$\exists T \mid \forall t > T, \forall i : x_i^t = x_{i+\omega}^t$$

When a configuration of a circular CA of size n at time t is spatially periodic with smallest period m, we shall indicate it as $X^t = (\alpha)^m$, where α is the sequence of values corresponding to the period. For example, the spatially periodic configuration $X = (a, b, a, b, a, b, a, b)$ can be written as $X = (a, b)^{n/2}$.

2.1.3 Orphans, Twins, and the Garden of Eden

It is possible for certain CA to have certain patterns (subsets of the global configuration) that have no predecessor. That is, it is impossible for a set of cells to assume a certain arrangement of states by the application of the local transition function on those cells. Such patterns are called *orphans*. An orphan cannot exist in any configuration except for the initial configuration.

It is possible for certain CA to have global configurations that can never be derived as the result of the application of the global transition function. Such configurations can only ever appear if they are defined as the initial configuration. Such states are called *Garden of Eden states*. A configuration is a Garden of Eden state if, and only if, it contains one or more orphans. This is in reference to the Biblical story of Eden, a location which was created out of nowhere, and could never be revisited once left.

As with any function, the global transition function of a CA may have the property of being injective, surjective, or bijective. We define these well known terms in the context of cellular automata below [31]:

A CA is *injective* if every global state is mapped to by the global transition function by at *most* one previous state. So, $F(X) = F(Y) \Rightarrow X = Y$.

A CA $F : C \rightarrow C$ is *surjective* if every global state is mapped to by the global transition function by at *least* one previous state. So, $\forall Y \in C, \exists X \in C$ such that $Y = F(X)$.

A CA is *bijective* if every state is mapped to by *exactly* one previous state. That is, a CA is bijective if it is both injective and surjective.

A bijective CA has the special property that it is always possible to know the previous state that came before the current state. We say that a bijective CA is *reversible* (Curtis-Hedlund-Lyndon theorem [23]). A reversible CA can be executed backwards in time from the current generation.

A *cyclic state* is a global state which will eventually be re-visited with the repeated application of the global transition function. States that are not cyclic are called *transient states*. Transient states can only be generated by the repeated application of the global transition function to a Garden of Eden State, and they can never be revisited. If all possible global configurations are cyclic, then the CA is reversible.

The Garden of Eden theorem states that a CA is injective if, and only if, it is surjective [43, 45]. More strongly, every non-injective CA has an orphan pattern, and therefore has a Garden of Eden state. As a corollary, every injective CA must also be surjective.

2.2 Elementary Cellular Automata

A one dimensional bi-infinite cellular automata with two states and a neighbourhood consisting of only itself and its left and right neighbours is known as an *elementary cellular automata* (ECA). There are exactly 256 possible ECA. ECA are significant because they represent the simplest possible case of what can be called a CA, making them accessible and simpler to understand. Certain properties that can be proven to exist in ECA, such as computation, can then be extended to more complex CA by means of induction.

For the most part, we will restrict our study to elementary cellular automata.

2.2.1 Canonical Forms of ECA

Binary

Neighbourhood		Transition
000	$\rightarrow$	r_0
001	$\rightarrow$	r_1
010	$\rightarrow$	r_2
011	$\rightarrow$	r_3
100	$\rightarrow$	r_4
101	$\rightarrow$	r_5
110	$\rightarrow$	r_6
111	$\rightarrow$	r_7

Table 2.1: Transition table

Elementary CA have only 8 possible local configurations, as shown above in table 2.1. We can express the local transition rule as a map from $\{0,1\}^3 \rightarrow \{0,1\}$:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (r_7, r_6, r_5, r_4, r_3, r_2, r_1, r_0)$$

The set of binary triplets on the left represent all possible inputs to the local transition function $f(x_1, x_2, x_3)$, where x_1, x_2, and x_3 represent the left, centre, and right neighbour, respectively. The set of binary numbers $(r_7, \ldots, r_0)$ represents the resultant values after applying the local transition function. These digits concatenated together are known as as the *binary representation* of the CA.

For example, if the elementary CA has the following transition map:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (1, 0, 0, 1, 1, 1, 0, 0)$$

then the binary representation of this transition rule is 10011100.

Decimal

The binary representation of the rule can converted to a decimal number in the standard way: $\sum_{i=0}^{7} 2^i r_i$. This decimal representation is known as the rule's *name* or *number*. Since there are only 8 possible inputs to the local transition function of an elementary CA, and the binary representation of an elementary CA only has 8 digits, the highest possible rule number is $2^8 - 1$, or 255.

For example, if we take the elementary CA above with the binary representation 10011100, its decimal representation (or name) is simply 156.

A one dimensional bi-infinite cellular automata with two states and a neighbourhood consisting of only itself and its left and right neighbours is known as an *elementary cellular automata* (ECA). There are exactly 256 possible ECA. ECA are significant because they represent the simplest possible case of what can be called a CA, making them accessible and simpler to understand. Certain properties that

can be proven to exist in ECA, such as computation, can then be extended to more complex CA by means of induction.

2.2.2 Empirical Classification

Wolfram observed that the 256 ECA seemed to produce four basic types of behaviour, regardless of their initial configuration. He classified them as follows (see section 3.2 for information on how to interpret these graphs):

Class 1 - Homogeneous

Class 1 ECA will eventually converge to a homogeneous configuration. This includes trivial rules such as rule 0, which always converge immediately to a configuration of all zeros. An example of class 1 is ECA rule 204 (fig. 2.1).

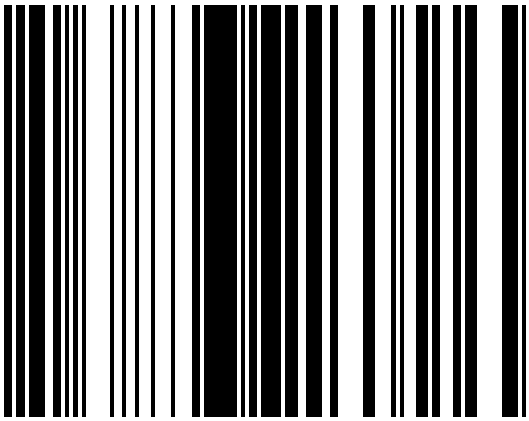


Figure 2.1: Rule 204

Class 2

Class 2 ECA will eventually converge to a periodic configuration. An example of class 2 is ECA rule 24 (fig. 2.2).

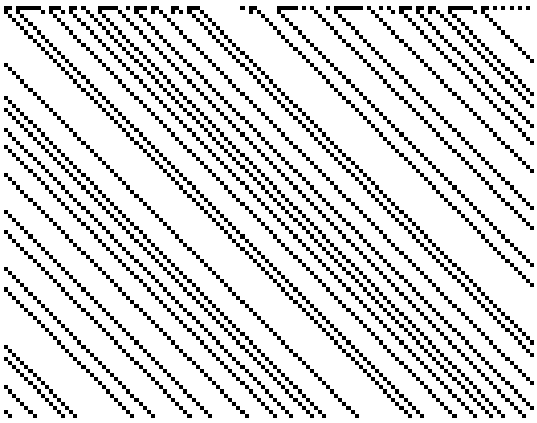


Figure 2.2: Rule 24

Class 3

Class 3 ECA evolve into aperiodic patterns that appear chaotic. As $t \to \infty$, the density of class 3 ECA tends toward a fixed value, regardless of its initial condition. An example of class 3 is ECA rule 30 (fig. 2.3).

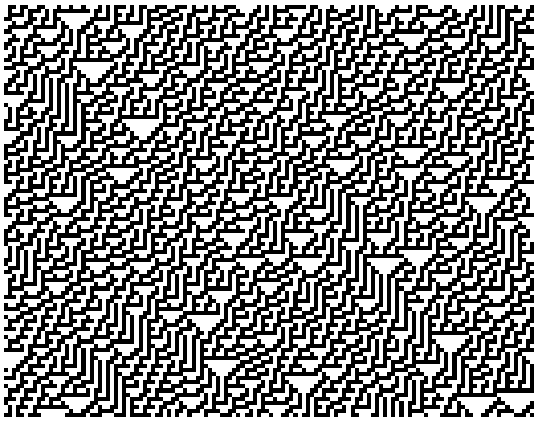


Figure 2.3: Rule 30

Class 4

Class 4 ECA produce interesting behaviour that can be described qualitatively as both ordered and random. Simple localized structures appear and interact with

each other in complex ways. An example of class 4 is ECA rule 110 (fig. 2.4).

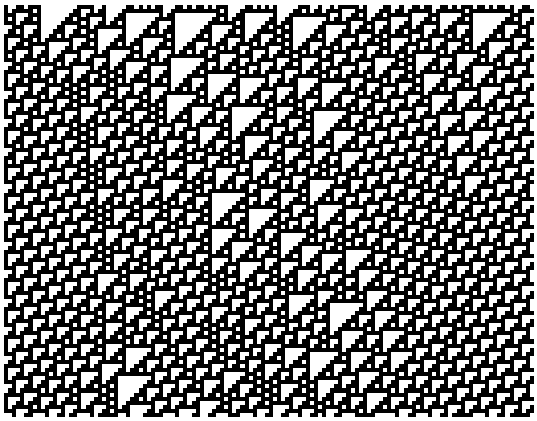


Figure 2.4: Rule 110

2.2.3 Normal Forms

The binary and decimal representations allow us to represent an elementary CA's local transition function in a manner that is concise and compact. However, it is sometimes necessary to represent a CA using standard function notation using a boolean expression. To represent a CA as a boolean function, we use a canonical or *normal* logical form. All canonical forms of Boolean functions are logically equivalent.

Disjunctive Normal Form (DNF)

The disjunctive normal form (DNF) consists of a disjunction of conjunctions of literals. An example of an expression in DNF is:

$$(A \wedge B) \vee (\neg A \wedge C)$$

We now wish to represent an elementary CA in DNF. Let us denote by b the

binary representation of the rule taken from the map:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (r_7, r_6, r_5, r_4, r_3, r_2, r_1, r_0)$$

Let us denote by b_i the i^{th} digit from the right of b (counting from zero). Let us denote by d_i the tuple mapping to b_i. Finally, let us denote by $d_{i,j}$ the j^{th} digit of d_i from the right (counting from one). The DNF of a boolean elementary CA is then expressed canonically as:

$$f(x_1, x_2, x_3) = \bigvee_{i|b_i=1} \bigwedge_{j=1:3} x_j^{d_{ij}}$$

where x^0 represents $\neg x$, and x^1 represents x.

For example, suppose we wish to find the DNF expression for rule 200. First, we begin with the rule map:

Neighbourhood		Transition	Binary Place
000	$\rightarrow$	0	r_0
001	$\rightarrow$	0	r_1
010	$\rightarrow$	0	r_2
011	$\rightarrow$	1	r_3
100	$\rightarrow$	0	r_4
101	$\rightarrow$	0	r_5
110	$\rightarrow$	1	r_6
111	$\rightarrow$	1	r_7

Table 2.2: Transition table for rule 200

We use this table to find the binary representation $b = 11001000$. The formula first requires us to find $i|b_i = 1$. We see that $b_i = 1$ when $i = 3, 6, 7$. Note that since there are three 1s in b, our final expression will be a disjunction of three clauses.

To find the first clause, we find d_3, the tuple mapping to b_3, which is $(0, 1, 1)$.

(Note that 011 is 3 in binary.) We then apply $\bigwedge_{j=1:3} x_j^{d_{ij}}$ which gives us the clause $(x_1^0 \wedge x_2^1 \wedge x_3^1)$, which is then written as $(\neg x_1 \wedge x_2 \wedge x_3)$.

We apply the same procedure to find the other two clauses. $d_6 = (1, 1, 0)$, which generates the clause $(x_1 \wedge x_2 \wedge \neg x_3)$. $d_7 = (1, 1, 1)$, which generates the clause $(x_1 \wedge x_2 \wedge x_3)$.

Finally, we disjunct the three clauses together to create the final expression, and write it in function notation:

$$f_{200}(x_1, x_2, x_3) = (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3)$$

Conjunctive Normal Form (CNF)

The conjunctive normal form (CNF) consists of a conjunction of disjunctions of literals. An example of an expression in CNF is:

$$(A \vee B) \wedge (\neg A \vee C)$$

We now wish to represent an elementary CA in CNF. We use the same denotations from the DNF case above. Again, we denote by b the binary representation of the rule taken from the map:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (r_7, r_6, r_5, r_4, r_3, r_2, r_1, r_0)$$

The CNF of a boolean elementary CA is then expressed canonically as:

$$f(x_1, x_2, x_3) = \bigwedge_{i|b_i=0} \bigvee_{j=1:3} x_j^{1-d_{ij}}$$

where x^0 represents $\neg x$, and x^1 represents x.

For example, suppose we wish to find the CNF expression for rule 233. First, we begin with the rule map:

Neighbourhood		Transition	Binary Place
000	$\rightarrow$	1	r_0
001	$\rightarrow$	0	r_1
010	$\rightarrow$	0	r_2
011	$\rightarrow$	1	r_3
100	$\rightarrow$	0	r_4
101	$\rightarrow$	1	r_5
110	$\rightarrow$	1	r_6
111	$\rightarrow$	1	r_7

Table 2.3: Transition table for rule 233

We use this table to find the binary representation $b = 11101001$. The formula first requires us to find $i|b_i = 0$. We see that $b_i = 0$ when $i = 1, 2, 4$. Note that since there are three 0s in b, our final expression will be a conjunction of three clauses.

To find the first clause, we find d_1, the tuple mapping to b_1, which is $(0, 0, 1)$. (Note that 001 is 1 in binary.) We then apply $\bigvee_{j=1:3} x_j^{1-d_{ij}}$ which gives us the clause $(x_1^1 \vee x_2^1 \vee x_3^0)$, which is then written as $(x_1 \vee x_2 \vee \neg x_3)$.

We apply the same procedure to find the other two clauses. $d_2 = (0, 1, 0)$, which generates the clause $(x_1 \vee \neg x_2 \vee x_3)$. $d_4 = (1, 0, 0)$, which generates the clause $(\neg x_1 \vee x_2 \vee x_3)$.

Finally, we conjunct the three clauses together to create the final expression, and write it in function notation:

$$f_{233}(x_1, x_2, x_3) = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

2.3 Fuzzy Cellular Automata

So far we have examined CA in the domain of Boolean logic; values are either *true* or *false*, denoted 1 or 0, respectively. We may wish to extend the notion of Boolean logic to include varying degrees of truth, so that a value can be "nearly true", or "mostly false", or "half true". Such an extension is known as *fuzzy logic*. With fuzzy logic, we retain our denotation of 1 being true and 0 being false. We extend these values by including values *between* 0 and 1. This is known as *many-valued logic*. In fuzzy logic, we denote the level of truth by choosing a value from the set $\{x|0 \leq x \leq 1, x \in \mathbb{R}$. Note that although this set is limited to the interval [0, 1], there are an infinite number of values between 0 and 1 from which to choose. This places fuzzy logic in stark contrast to Boolean logic, where a value may have only two possible states.

A *fuzzy cellular automaton* (FCA) is a cellular automaton whose cells may have real number values in the range of $[0, 1]$, rather than being restricted to values from the binary set $\{0, 1\}$. In a FCA, the state set is $S = [0, 1]$. They represent a "fuzzification" of the Boolean behaviour of a corresponding Boolean CA. It is worth noting that while the canonical forms of DNF and CNF are logically equivalent in the Boolean case, they are not necessarily equivalent in the fuzzy case.

Since we are now using real numbered values instead of Boolean values, we can no longer use Boolean operators in our equations. We must "fuzzify" the Boolean

operators, which consists of replacing them with standard algebraic operators. The method by which we replace Boolean operators is known as *a logic*. There are many different fuzzy logics available, and each attempts to preserve certain properties of their Boolean counterparts. One property that is generally maintained by all logics is that a fuzzy logic should produce the same result as standard Boolean logic when operating on Boolean values. Some of the more common fuzzy logics used in CA are presented below in table 2.4:

Logic	$\neg \boldsymbol{x}$	$\boldsymbol{x} \wedge \boldsymbol{y}$	$\boldsymbol{x} \vee \boldsymbol{y}$
CFMS	$1 - x$	$x \cdot y$	$min\{1, x + y\}$
Probabilistic	$1 - x$	$x \cdot y$	$x + y - x \cdot y$
Łukasiewicz	$1 - x$	$max\{0, x + y - 1\}$	$min\{1, x + y\}$
Gödel	if $x = 0$ then 1, else 0	$min\{x, y\}$	$max\{x, y\}$
Zadeh	$1 - x$	$min\{x, y\}$	$max\{x, y\}$
Product	if $x = 0$ then 1, else 0	$x \cdot y$	$x + y - x \cdot y$

Table 2.4: Fuzzy Logics [42]

Fuzzy logic must maintain certain properties of its Boolean counterpart. The most important property in a fuzzy logic is that it maintains the rules of Boolean logic when operating in a boolean space. In other words, when presented with Boolean data, fuzzy logic operators should produce the same results as their Boolean counterparts. This is true of all the logics presented in table 2.4. Since fuzzy logic operates on continuous values, it is also desirable for fuzzy logic functions to be continuous as well. This is true for all logics shown in table 2.4, with the exception of Gödel and Product logics. It may also be possible to show the preservation of other mathematical properties under fuzzification. In [5], it was shown that for elementary fuzzy CA under CFMS in DNF, there was an equivalence between convergent fuzzy CA and the mean field approximation on Boolean CA, as well as the preservation of the CA's density conservation.

To convert a Boolean CA into a fuzzy CA, we first convert it into one of the normal forms from the previous sections, then we convert the Boolean operators to standard operators by choosing one of the logics from the above table.

For example, to find the formula for elementary fuzzy CA rule 200 in disjunctive normal form using CFMS logic, we first convert Boolean rule 200 into DNF as described in section 2.2.3:

$$f_{200}(x_1, x_2, x_3) = (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3)$$

Then we apply the logic from table 2.4 to the above formula, which produces:

$$f_{200}(x_1, x_2, x_3) = min\{1, ((1 - x_1) \cdot x_2 \cdot x_3) + (x_1 \cdot x_2 \cdot (1 - x_3)) + (x_1 \cdot x_2 \cdot x_3)\}$$

This can then be simplified using standard algebra to

$$f_{200}(x_1, x_2, x_3) = min\{1, (x_1 \cdot x_2 + x_2 \cdot x_3 - x_1 \cdot x_2 \cdot x_3)\}$$

although the simplification step may sometimes be omitted when not required, such as during a computer simulation. Note that we restrict the result to ≤ 1 by using the *min* function. It was shown in [20] that this was not necessary with DNF fuzzification, as no terms ever exceeded 1.

For another example, to find the formula for elementary fuzzy CA rule 233 in conjunctive normal form using Łukasiewicz logic, we first convert Boolean rule 233

into CNF as described in section 2.2.3:

$$f_{233}(x_1, x_2, x_3) = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

Then we apply the logic from table 2.4 to the above formula, which produces:

$$\begin{aligned} f_{233}(x_1, x_2, x_3) = & \, max\{0, min\{1, (x_1 + x_2 + (1 - x_3))\} \\ & + max\{0, min\{1, (x_1 + (1 - x_2) + x_3)\} \\ & + min\{1, ((1 - x_1) + x_2 + x_3)\} - 1\} - 1\} \end{aligned}$$

Further simplification becomes very difficult without having some knowledge of the values of x_1, x_2, x_3.

2.4 Symbolic Dynamics and Chaos

2.4.1 Symbolic Dynamics

Symbolic dynamical systems were introduced in [44] as a method of representing a topological dynamical system as an infinite sequence of a finite set of symbols. They are useful in that they can provide a conjugacy between a continuous topology and a discrete system.

The standard space used in symbolic dynamics is the sequence space: $\Sigma_n =$

$\{(s_0 s_1 s_2 \ldots) | s_j \in S\}$, where S is a finite set of symbols, usually nonnegative integers. The smallest possible size of S is 2. This is known as the *binary sequence space*:

Definition 2.1: The *binary sequence space* is the set $\Sigma_2 = \{(s_0 s_1 s_2 \ldots) | s_j = 0 \text{ or } 1\}$.

When the size of the symbol set is not given, it can be assumed to be binary, and the binary sequence space can be denoted simply as Σ. In order to analyze the dynamics of points in Σ, we must first define a metric that can quantitatively compare any two points in that set.

Definition 2.2: A function d is called a *metric* on X if for any $x, y, z \in X$, the following three properties hold:

1. $d[x, y] \geq 0$, and $d[x, y] = 0$ if and only if $x = y$
2. $d[x, y] = d[y, x]$
3. $d[x, z] \leq d[x, y] + d[y, z]$

The metric most commonly used to compare points in Σ is the distance metric:

Definition 2.3: Let $s = (s_0 s_1 s_2 \ldots)$ and $t = (t_0 t_1 t_2 \ldots)$ be two points in Σ. The *distance* between s and t is given by:

$$d[s, t] = \sum_{i=0}^{\infty} \frac{|s_i - t_i|}{s^i}$$

The metric d represents the distance between two points in Σ. The following proximity theorem shows how the distance between two points manifests itself in the sequences:

Theorem 2.1: Let $s, t \in \Sigma$ and suppose $s_i = t_i$ for $i = 0, 1, \ldots, n$. Then $d[s, t] \leq \frac{1}{2^n}$. Conversely, if $d[s, t] < \frac{1}{2^n}$, then $s_i = t_i$ for $i \leq n$.

Using the distance metric defined previously for Σ, the proximity theorem dictates that two sequences are within a distance of $\frac{1}{2^n}$ of each other if their first n entries agree. So, two sequences are "close" if their initial entries are the same.

Finally, we introduces the *shift map* function on Σ. The *shift map* $\sigma : \Sigma \to \Sigma$ is defined by:

$$\sigma(s_0 s_1 s_2 \ldots) = (s_1 s_2 s_3)$$

So, the shift map literally shifts all symbols to the left, discarding the initial symbol.

2.4.2 Chaos

There is no single and universally accepted definition of chaos. For our purposes, we will use the definition proposed by Robert Devaney [16] [17]:

Definition 2.4: A dynamical system F is chaotic if:

1. Periodic points for F are dense.
2. F is transitive
3. F depends sensitively on initial conditions

The above definition relies on the following definitions:

Definition 2.5: Let X be a set, and let $Y \subset X$. Y is *dense* in X if, for any point

$x \in X$, there is a point $y \in Y$ arbitrarily close to x. Note: closeness is defined by a metric or distance function on X.

So, for example, the subset of rational numbers is dense in the set of real numbers, since any real number (including irrational numbers) can be approximated to any arbitrary precision with a rational number.

Definition 2.6: A dynamical system is *transitive* if for any pair of points x, y, and any $\epsilon > 0$, there is a third point z within ϵ of x whose orbit comes within ϵ of y.

In other words, given any two points, an orbit exists that comes arbitrarily close to both.

Definition 2.7: A dynamical system F *depends sensitively on initial conditions* if there is a $\beta > 0$ such that for any x and any $\epsilon > 0$ there is a y within ϵ of x and a k such that the distance between $F^k(x)$ and $F^k(y)$ is at least β.

In other words, given a point x, you can always find a point y within an arbitrary distance of x whose orbit eventually separates from x by at least β.

Example 2.1: Show that σ on Σ is chaotic:

It is sufficient to show:

1. Sensitivity to initial conditions
2. Density of periodic orbits
3. Topological mixing

We demonstrate each of these properties below:

Density of periodic orbits

Let $\hat{s} = (0\,1\ 00\,01\,10\,11\ 000\,001\ \ldots)$. That is, $\hat{s}$ is the sequence which consists of all possible clocks of 0s and 1s of length 1, followed by all such blocks of length 2, then length 3, etc.

Now, choose any arbitrary point $s = (s_0 s_1 s_2 \ldots) \in \Sigma$, choose $\epsilon > 0$, and choose n so that $\frac{1}{2^n} < \epsilon$. By definition, at some position k in $\hat{s}$ lies the digits $s_0 s_1 s_2 \ldots s_n$. If you apply the shift map σ k times to $\hat{s}$, then the first $n+1$ entries of $o^k(\hat{s})$ are $s_0 s_1 s_2 \ldots s_n$. By the Proximity Theorem,

$$d[\sigma^k(\hat{s}), s] \leq \frac{1}{2^n} < \epsilon$$

Therefore, the point $\hat{s}$ has an orbit that forms a dense subset of Σ.

Sensitivity to initial conditions

Let $\beta = 1$. For any $s \in \Sigma$ and $\epsilon > 0$, choose n so that $\frac{1}{2^n} < \epsilon$. Suppose $t \in \Sigma$ satisfies $d[s,t] < \frac{1}{2^n}$ but $t \neq s$. Then we know that $t_i = s_i$ for $i = 0, \ldots, n$. However, since $t \neq s$, there exists $k > n$ such that $s_k \neq t_k$. So, $|s_k - t_k| = 1$. Now consider the sequences $\sigma^k(s)$ and $\sigma^k(t)$. The initial entries of each of these sequences are different, so we have

$$d[\sigma^k(s), \sigma^k(t)] \geq \frac{|s_k - t_k|}{2^0} + \sum_{i=0}^{\infty} \frac{0}{2^i} = 1$$

.

Therefore, the shift is sensitive to initial conditions.

Topological mixing

A dynamical system is *transitive* if for any pair of points x and y, and any $\epsilon > 0$,

there is a third point z within ϵ of x whose orbit comes within ϵ of y. Clearly, since Σ has an orbit that forms a dense subset of Σ, then Σ is topologically mixing, since $\hat{s}$ comes arbitrarily close to all points.

All three criteria for chaos have been met, therefore the shift map σ on Σ is chaotic.

2.5 Conclusion

In this chapter, we have introduced the definitions needed in the thesis. In particular, we have introduced the definition of Boolean cellular automata, we have described one of the most common classification of their behaviour employed in the literature, we have shown the different ways of representing elementary cellular automata through disjunctive and conjunctive normal forms, we have then described their extension to Fuzzy cellular automata by different "fuzzification" of the normal forms. Finally, we have introduced the definition of chaos in dynamical systems.

Chapter 3

Simulation and Visualization

As a prerequisite to our study of cellular automata, we created robust and flexible software called CellView to facilitate our study. While software packages such as Mathematica can simulate and visualize CA, we decided that software tailored specifically for CA would be desirable.

In this chapter, we introduce some of the features of our software, focusing specifically on the simulation and visualization features that are important to CA research in general.

3.1 Simulation

3.1.1 Features

Our software is built from scratch in Java using open source libraries (see *"Software and Media Licences"* in the appendix). The user interface is shown in figure 3.1.

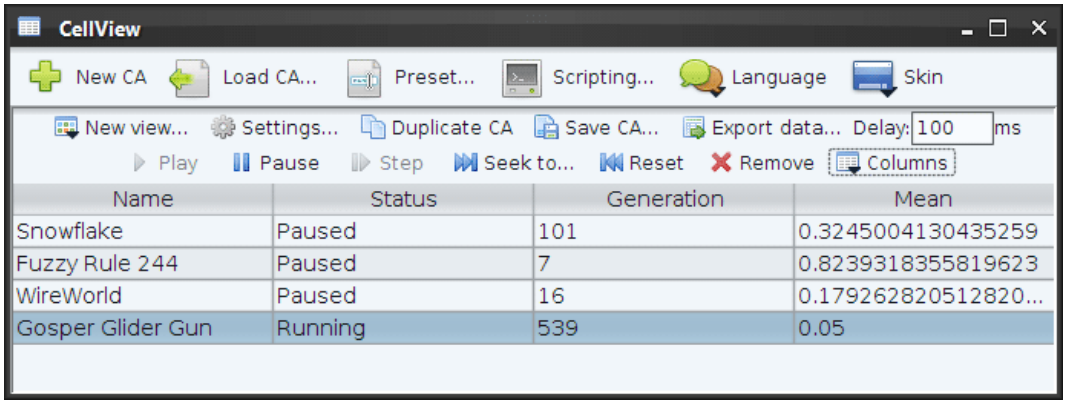


Figure 3.1: CellView main window

With our software, users can create, save and load multiple CA rule simulations. The execution of these simulations is controlled with familiar controls such as *Play*, *Pause*, etc. The speed of execution can be controlled, and rules can also be executed one step at a time. It is also possible to seek to any specified step. Multiple rules can be simulated, and can also be synchronized to each other. Data such as the current minimum, maximum, variance, standard deviation, etc., can be displayed for each rule, and the raw data of the current configuration can be exported for further examination. Multiple visualization windows can be opened for each rule (see section 3.2).

3.1.2 Scripting and Compiling

There are many possible methods of representing and executing a CA's local transition function with a computer. Several options were considered, including expression evaluators, which interpret standard algebraic expressions. Expression evaluators are, however, very limiting. Many complex CA rules, such as the FCA snowflake (section 4.3.2), are more easily expressed algorithmically, rather than algebraically. For this reason, we chose to create an algorithmic CA evaluator.

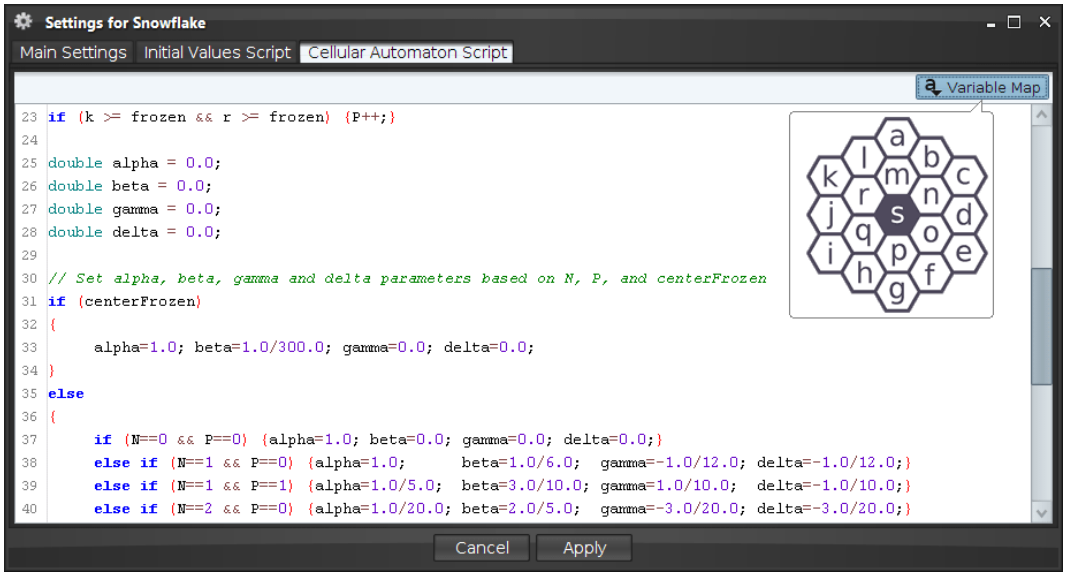


Figure 3.2: CA Script for Hexagonal Snowflake

Perhaps the most straightforward method of executing an algorithmic CA is to code the desired behaviour as a function or a method, and compile these definitions into the program. This is not a robust solution, as it requires the entire program to be altered and recompiled each time a new CA is to be studied.

We have instead decided to implement a system that allows CA algorithms to be compiled and executed on-the-fly, using the Janino compiler library for Java [52]. The local transition function is represented as a script. The script is essentially the inside of a Java method. The local neighbours are passed as parameters to this method, and a graphical display of these neighbourhood variables are presented to the user (see figure 3.2). The method must end by returning a double-precision floating point value. This is the result of the local transition function.

When the simulation is started, the script is compiled to Java byte code, and executed on the same virtual machine as the main program. This differs from other systems which interpret scripts in real time rather than compiling and executing them. We also employ multi-threading to improve performance. In our informal tests, this

method affords a significant performance benefit over interpreted scripting. In addition, this system enables the user to access not only all of Java's standard libraries, but to import any external libraries that are desired. For example, the user could use an external random number generator rather than Java's internal one.

Figure 3.3 demonstrates the script for FCA rule 244. The variables *a, b,* and *c* represent the left, centre, and right neighbours, respectively (this is presented to the user as a diagram). Note that simple algebraic expressions, such as this one, can be represented as a single *return* statement.

Figure 3.4 demonstrates the Game of Life [21] defined as a multi-statement algorithm. In the Game of Life, the state of the cell depends on its own value, and the sum of its neighbours. This is an example of how a Boolean CA can easily be represented as a real-valued system.

```
return a*b*c
        + a*b*(1.0-c)
        + a*(1.0-b)*c
        + a*(1.0-b)*(1.0-c)
        + (1.0-a)*b*(1.0-c);
```

Figure 3.3: Script for FCA rule 244

```
double result = 0.0;
double nSum = b+c+d+h+l+k+j+f;
if (g==1.0 && (nSum==2||nSum==3))
        {result = 1.0;}
if (g == 0.0 && nSum == 3)
        {result = 1.0;}
return result;
```

Figure 3.4: Script for the Game of Life

The initial values of the CA are also defined with a similar script. This allows the user to define values manually, or to generate them procedurally (for example, to generate a set of random initial values). These scripts, along with the other settings such as the CA's name, dimension, size, neighbourhood radius, and background, can then be saved as an XML file for later use.

One important note must be made about the nature of this type of CA simulation. Using our program, it is possible for the user to create a system that violates

the constraints placed on CA. For example, it is possible to define a CA that uses networking or disk access libraries to allow cells to observe the value of cells beyond their local neighbourhood, or to take their values from some external source. The onus is on the user to ensure that their CA conforms to whatever constraints they are using.

3.1.3 Limitations

Digital computers are, by definition, both finite and discrete. These limitations have two important consequences when simulating FCA.

Computers are finite in that any physical computer has a finite memory, so it is impossible to truly simulate an infinite CA using a physical computer. Section 2.1.2 discusses two exceptional cases in which a finite CA can simulate an infinite CA. Both of these cases are supported with our software. In the first case, a spatially periodic infinite CA may be simulated as a circular CA. In the second case, a subset of an infinite CA may be simulated to a finite number of time steps by adding extra cells to the boundaries. This is supported in our software. The main settings window contains a numerical field called *"Background"*. Increasing this number from zero will add cells to the boundaries, but these cells will not be seen during visualization, nor will their values be included in in any of the statistical property displays such as minimum, maximum, or variance.

Computers are discrete in that each memory cell can only assume one of a finite number of states. This means that there is a limitation to the precision in which real-valued numbers can be stored. This effect is compounded in FCA simulations, as the global transition function is applied recursively, causing errors to amplify. Our program uses IEEE 754 double precision, 64-bit floating point numbers for all calcu-

lations [25]. In the future, we would like to give the user the ability to use any data type, such as an arbitrary precision floating point data type.

3.2 Visualization

(Note: Some images in this section may not be legible on a black and white copy of this document.)

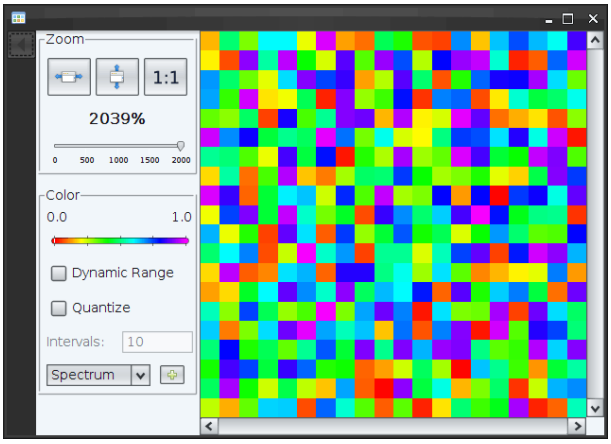


Figure 3.5: Visualization Window

3.2.1 Colour Maps

Nearly all CA visualization techniques involve mapping a cell's state space to a colour space. With Boolean CA, convention states that zero is mapped to white, and one is mapped to black. With fuzzy CA, no one standard colour mapping exists. Generally, the continuous values of FCA are mapped to a continuous colour gradient.

Our software allows the user complete flexibility in defining a colour gradient map. A colour legend is visible on all visualization screens, representing values in the range of $(0, 1)$. The user can add or remove colour points at any point on this legend.

Each colour point can be assigned a colour of the user's choosing. The program will then automatically interpolate the colours of all other points on the legend (fig. 3.6). In the simplest case of two colour points, the legend will fade between these two colours. However, the user may choose as many colour points as they like. We present some of the colour maps we have found useful below.

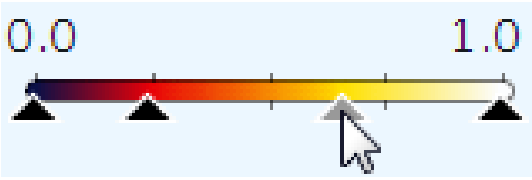


Figure 3.6: Colour Legend

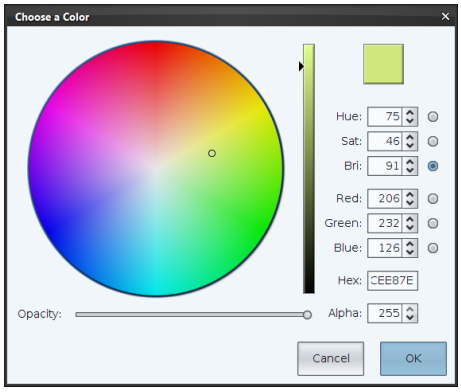


Figure 3.7: Colour chooser

a) white to black

b) probabilistic

c) spectrum

d) spectrum with white and black

Figure 3.8: Useful colour maps

White to Black

(fig. 3.8 a) This is a extension of the standard Boolean convention of zero being white and one being black. In this colour map, the degree of darkness correlates with the distance of that value to one. Note that Boolean values will appear as they would in the standard Boolean convention for colours.

Probabilistic

(fig. 3.8 b) In this colour map, all values less than 0.5 will appear white, and all values greater than 0.5 will appear black. This can be considered as an indication of whether the value is probably zero, or probably one. Once again, Boolean values will conform to the standard convention.

Spectrum

(fig. 3.8 c) The standard colour spectrum is often employed in legends whenever a continuous colour map is required. With FCA, this has the drawback that Boolean values will appear as red for zero, and violet for one, in violation of the standard convention. Red and violet also do not have much contrast, so this colour map is not suitable for FCA with many Boolean values.

Spectrum with White and Black

(fig. 3.8 d) In order to use the spectrum colour map, yet also conform to the standard convention for Boolean values, we place white and black colour points at the ends of the interval. This causes colours to tend toward white as they approach zero, and tend toward black as they approach one.

Spectrum with Black and White

This colour mapping is identical to the one above, but the black and white colours are reversed. That is, colours to tend toward black as they approach zero, and tend toward white as they approach one. This colour mapping was introduced specifically for the printing of this thesis, as many of our diagrams contain graphs of

CAs with many values equal to one. This is the colour mapping used for diagrams in all subsequent chapters.

3.2.2 Colour Options

Dynamic Range

Typically, FCA have the state space $S = [0, 1]$, so the colour gradient maps to this range. There are two situations where this may not be desirable. First, a FCA could be designed to have values beyond $[0, 1]$. Secondly, an FCA could have a configuration with values that exist within a tiny segment of $[0, 1]$. The latter is quite common when FCA tend to converge toward some value, without ever reaching it.

The dynamic range option alters the gradient map dynamically in real time. It ensures that the left side of the gradient is mapped to the minimum value in the current configuration, and the right side is mapped to the maximum value. The software updates the numerical labels in the legend to reflect the current range. (fig 3.9, 3.10)

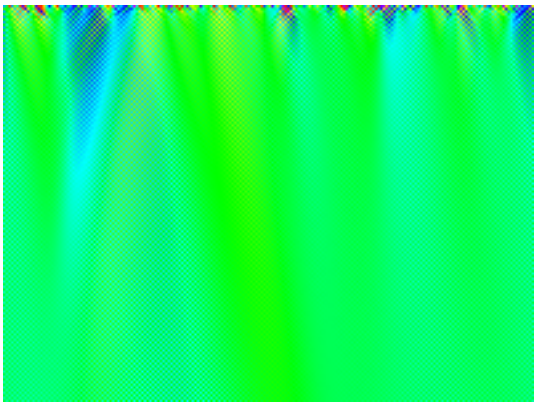


Figure 3.9: FCA rule 184 without dynamic range

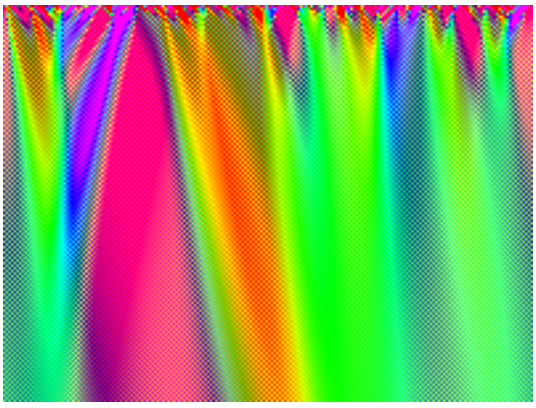


Figure 3.10: FCA rule 184 with dynamic range

Colour Quantization

Sometimes the smooth colour gradient can mask fine structures in the data. The colour quantization option reduces the total number of colours displayed to a value of the user's choosing. This can make certain hidden structures more visible. (fig 3.11, 3.12)

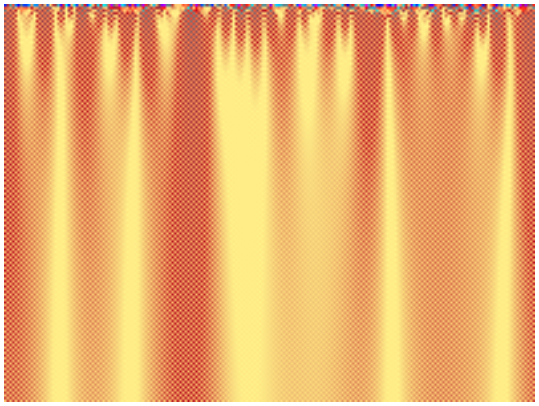


Figure 3.11: FCA rule 18 without colour quantization

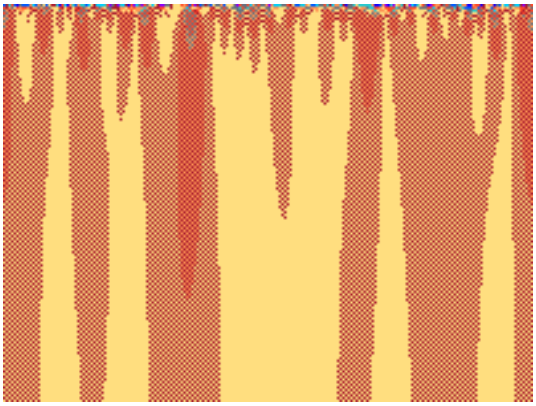


Figure 3.12: FCA rule 18 with 10-colour quantization

3.2.3 Views

Space View

Two-dimensional CA are visualized by displaying the current configuration as a grid of coloured squares. This view represents a snapshot of the CA at the current generation only, and this view will change with time as the automaton changes. In this way, it behaves as an animation. (fig 3.13, 3.14)

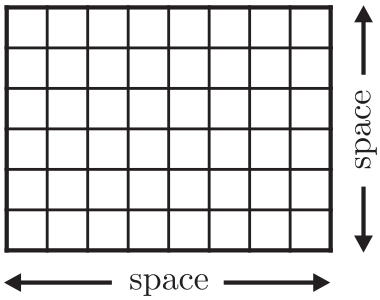


Figure 3.13: Space diagram

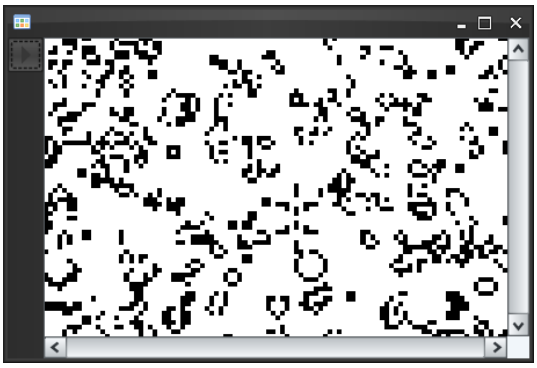


Figure 3.14: Space diagram of the Game of Life

Space-Time View

When displaying a one-dimensional CA on a two-dimensional screen, we can take advantage of this extra dimension by using the vertical dimension to represent time. By convention, the bottom row of the display represents the current generation, and the top of the screen represents the oldest generation that can be displayed (often we wish to have the initial configuration at the top). With each new generation, the previous generations are "pushed" upwards until they scroll off the screen (fig 3.15, 3.16). Note that diagrams in all subsequent chapters are of this type.

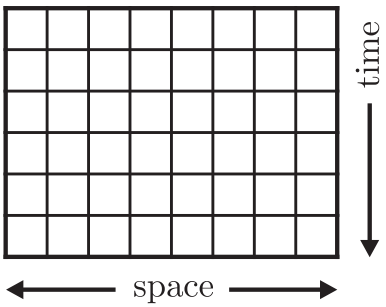


Figure 3.15: Space-time diagram

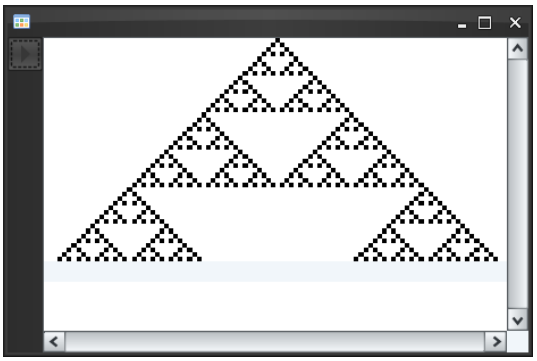


Figure 3.16: Space-time diagram of rule 90

Radial View

The radial view was introduced in [20] as a new method of visualizing one-dimensional FCA. In the radial view, the linear array is wrapped around a circle. Each value is assigned a point defined in polar co-ordinates whose angle about the origin depends on its position in the array. The radius of each point from the origin depends on the value. So, values of zero will always appear at the origin, while values of one will always appear as a point on the unit circle. The user has the option to connect the points to form a curve. (fig. 3.17) We also preserve the colour mapping while drawing these curves.

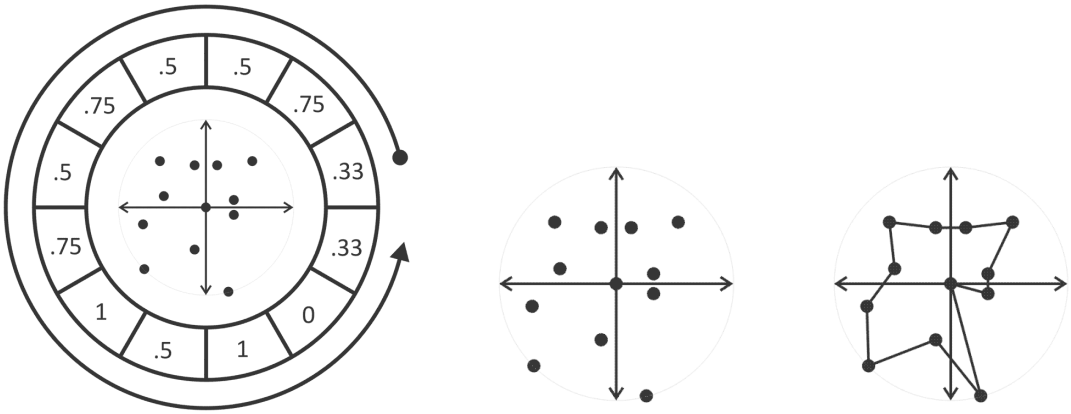
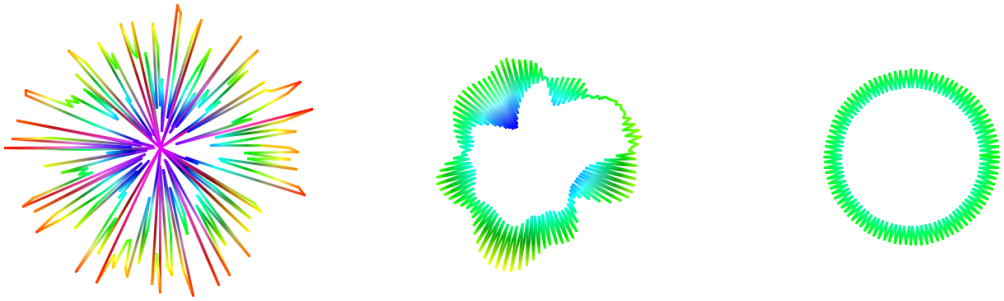


Figure 3.17: Radial view

The radial view can sometimes reveal structures that are not easily seen with the space-time diagram. In particular, it exposes spatial correlations among cells that may not be adjacent to each other. Observe the radial evolution of rule 184. (fig. 3.18) In rule 184, every second cell is spatially correlated. In the radial view, this appears as two distinct curves, that eventually converge to concentric circles.



a) Initial configuration b) After 50 generations c) After 5500 generations

Figure 3.18: Radial view of FCA rule 184

Bipartite and Weighted View

We introduce two new forms of CA visualization: the bipartite and weighted graph views. These views are only available for two-dimensional CA with square dimensions (identical width and height). With these views, the CA grid is interpreted as a weighted incidence matrix. That is, cell at row i and column j represents the weight of the edge from vertex i to vertex j in a directional graph. Clearly, an $m \times m$ CA will have a graph with m nodes. This incidence matrix is then visualized in two ways. In each case, vertices are drawn as points, and edges as line segments.

In the bipartite case, each vertex is shown twice, in two identical columns (fig. 3.19). An edge from vertex i to vertex j is represented as a line connecting vertex i on the left with vertex j on the right.

In the weighted graph view, each vertex is drawn as equidistant points arranged in a circle. An edge from vertex i to vertex j is drawn as an arc, and the direction is indicated with an arrow head.

In both cases, the same colour options are available as in other views. In addition, the opacity of the lines also indicate their value. A weight of one is opaque, while a weight of zero is completely transparent (invisible). Without this feature, the weighted

graph would always appear as a complete graph.

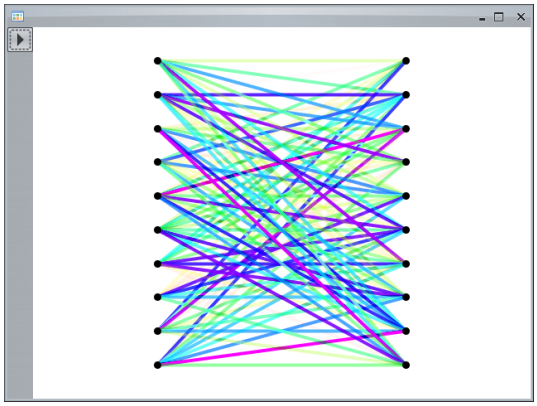


Figure 3.19: Bipartite graph

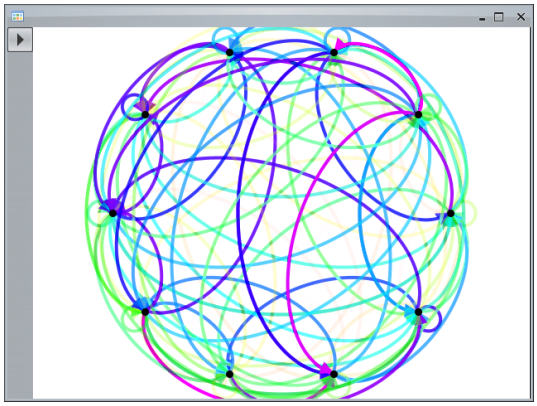


Figure 3.20: Weighted graph

3.3 Conclusion

Having a tool such as CellView has been useful in our investigation, and we hope it will be useful to others as well. The most important information from this chapter that is relevant to the rest of this thesis is that all subsequent diagrams utilize the *"Spectrum with Black and White"* colour mapping from section 3.2.1, and the space-time diagram layout from section 3.2.3.

Chapter 4

Fuzzy CA in Disjunctive Normal Form

In this chapter, we define how FCA in DNF have been classified, and we discuss the relationship between FCA and Boolean CA. Finally, we demonstrate some applications of FCA.

Recall that a *fuzzy cellular automaton* (FCA) is a cellular automaton whose cells have the state space of $S = [0, 1]$. They represent a "fuzzification" of the Boolean behaviour of a corresponding Boolean CA, usually in disjunctive normal form (see sections 2.3 and 2.2.3). In this chapter, we review previous research that has been done on analyzing fuzzy CA in DNF.

Initial investigations of FCA in DNF demonstrated that they had very complex behaviour. It was thought that perhaps this complex behaviour was a manifestation of chaos (see section 2.4.2). It was later shown by A. Mingarelli that fuzzy CA do not, in fact, exhibit chaos [41, 42].

FCA in DNF still appeared to exhibit interesting dynamics, so subsequent work focused on defining and classifying the different behaviours observed. Initial work on

the classification of circular FCA was done empirically, which is discussed below in section 4.1.1. This work was later refined analytically, which is discussed below in section 4.1.2.

4.1 Classifications

4.1.1 Empirical Classification

Circular elementary fuzzy cellular automata in the disjunctive normal form were studied empirically by Flocchini and Cezar by observing computer simulations evolved from a random initial configuration [20]. They observed that all FCA in DNF appear to converge toward an asymptotic periodic behaviour. Furthermore, all rules appear to have a temporal period of 1, 2, 4, or n. It is conjectured that all circular elementary FCA rules in DNF are indeed asymptotically periodic. This does not correspond to the behaviour seen in circular boolean CA.

Table 4.1 lists elementary rules and their periods (rules that are equivalent under transformations such as conjugation and reflection are omitted). We discuss certain cases below.

Homogeneous Configurations

Most FCA rules in DNF appear to become homogeneous in time. For all cases, the homogeneous configuration reached in the simulation is truly its homogeneous fixed point configuration (i.e. $F(X) = X$, where $X = (x, x, \ldots, x)$). Many of the FCA reach a homogeneous configuration that is quiescent state, where $X = (0, 0, \ldots, 0)$, although they do so at different speeds (some require a great number of iterations to

Period	Rules
Period 1 (Quiescent)	R_0, R_8, R_{32}, R_{40}, R_{72}, R_{104}, R_{128}, R_{136}, R_{160}, R_{168}, R_{24}, R_{36}, R_{152}, R_{164}, R_{44}, R_{56}, R_{74}, R_{200}
Period 1 (Homogeneous)	R_6, R_9, R_{22}, R_{25}, R_{26}, R_{30}, R_{33}, R_{35}, R_{37}, R_{38}, R_{41}, R_{45}, R_{54}, R_{57}, R_{60}, R_{61}, R_{62}, R_{73}, R_{90}, R_{105}, R_{106}, R_{110}, R_{122}, R_{126}, R_{134}, R_{150}, R_{154}, R_{172}
Period 1 (Heterogeneous)	R_4, R_{12}, R_{13}, R_{76}, R_{77}, R_{132}, R_{140}, R_{204}, R_{232}, R_{28}, R_{108}, R_{156}
Period 1 for n even (Heterogeneous)	R_{78}, R_{94}
Period 2 for all n	R_1, R_5, R_{19}, R_{23}, R_{27}, R_{50}, R_{51}, R_{178}
Period 2 for n even	R_{18}, R_{29}, R_{58}, R_{146}, R_{184}
Period 4 for n multiple of 4	R_{46}
Period n (Shifts)	R_2, R_{10}, R_{15}, R_{34}, R_{42}, R_{130}, R_{138}, R_{162}, R_{170}, R_3, R_7, R_{11}, R_{14}, R_{43}, R_{142}

Table 4.1: Observed dynamics of FCA in DNF

converge, while others require few). Other rules reach a homogeneous configuration other than zero. In all cases, this requires few iterations.

Heterogeneous Fixed-Point Configurations

Certain FCA rules in DNF seem to converge to a configuration that is spatially heterogeneous, yet temporally homogeneous (i.e. a periodic non-homogeneous configuration, with period 1). For some rules, such as $R_{204}(f(x, y, z) = y$, the reason for this behaviour is obvious, since the radius of the local transition function is effectively 0, and each value is copied exactly from the previous iteration.

Two rules, R_{94} and R_{78}, appear to converge to a homogeneous fixed point when its size is odd, while converging to a periodic configuration when its size is even. R_{94}

converges toward the configuration $X = (a, b)^{\frac{n}{2}}$, where $2a + 2b - ab = 2, a \neq b$. R_{78} converges toward the configuration $X = (a, b)^{\frac{n}{2}}$, where $a + b = 1, a \neq b$.

Periods of Length Two

An obvious example of of a FCA with a period of 2 is $R_{51}(f(x, y, z) = 1 - y))$, where each cell will become its fuzzy logic complement with each iteration. Other rules demonstrate a 2-periodic behaviour, but require several iterations to stabilize into this pattern. One rule, R_{27} requires some time to converge toward periodic behaviour. Once it does, its behaviour is easily explained, as the local transition function simplifies to $f(x, x, x) = 1 - a$ in the homogeneous case.

Several rules appear 2-periodic, but only when n is even. While this is perplexing at first, further investigation reveals the reason for this behaviour. These rules are actually n-periodic, i.e. they are shifting. A shifting configuration will appear 2-periodic in time if they are also 2-periodic in space. The rules in this class will converge to a spatial configuration of $X = (a, b)^{\frac{n}{2}}$, at which point their shifting behaviour appears 2-periodic. This spatially periodic configuration can only occur when the size of the CA is even. When the size is not even, these rules will converge toward a homogeneous fixed-point configuration.

Periods of Length Four

There is only one rule with a period of four: R_{46}. As in the case above, this periodic behaviour is actually a shift of a spatially periodic configuration $X = (a, b, 1 - a, 1 - b)^{\frac{n}{4}}$, and as such, only appears when the size of the CA is a multiple of 4.

Periods of Length n

As discussed above, some rules exhibit shifting behaviour. A simple example is $R_{170}(f(x, y, z) = z)$, which causes any configuration to shift to the left at each iteration. Other rules exhibit shifting behaviour only after several iterations. More complex shifts can occur, such as $R_{15}(f(x, y, z) = 1 - x)$, where the configuration is complemented as it shifts, so that every other configuration is shifted by two positions (called a double alternating shift).

4.1.2 Analysis of Empirical Classification

While simulation is a good method of observing the behaviour of FCA, simulation is not sufficient proof that these properties hold true in all cases. There are an infinite number of possible random initial configurations, and each simulation run is limited by the finite precision of storage and calculation.

An analytical proof of the asymptotic properties is desirable, but in their standard form, FCA are difficult to analyze mathematically. Betel and Flocchini proved the asymptotic behaviour of certain classes of FCA in DNF [5], namely: weighted average CA, and self-averaging CA.

In the case of weighted average CA, the analysis consists of four steps. First, the concept of a *weighted average* is defined. Secondly, this concept is applied to define a new type of CA called a *generalized fuzzy CA with weighted average*. Third, generalized FCA are shown to be convergent. Finally, certain FCA rules in DNF are shown to be equivalent to a generalized FCA. We elaborate briefly on each of these steps below.

Weighted Average

A weighted average of two numbers is similar to the concept of a *mean* average, but one value is given more weight than another. For two numbers, α and β, the weighted average μ is equal to $\mu = (1-\gamma)\alpha + \gamma\beta$ where $\gamma \in [0,1]$. Note the weighted average becomes the mean when $\gamma = \frac{1}{2}$.

Generalized CA

By definition, a cellular automaton is defined as a regular lattice of anonymous cells with identical local transition functions. We now extend this definition, however, to include CA where the local transition function differs from cell to cell. This type of CA is called a *generalized cellular automaton.*

Generalized FCA with Weighted Average

We now apply the definitions from the two previous subsections and define a type of CA called a *generalized fuzzy cellular automata with weighted average* (GWCA), which has the following form:

$$x_i^{t+1} = \gamma_i^t x_i^t + (1-\gamma_i^t)x_{i+1}^t$$

with bounded weights. That is, there exists $0 < \gamma < \frac{1}{2}$ such that $\gamma_i^t \in (\gamma, 1-\gamma)$ for all i and for all t. So, the state of a cell at time $t+1$ is the average of itself and its right neighbour at time t, weighted by a value in $(0,1)$ that varies from cell to cell. Note that a similar definition exists for rules averaged with the left neighbour.

General Convergence Theorem

The general convergence theorem states:

For a GWCA with starting configuration $X^0 = (x_0^0, \ldots, x_{n-1}^0)$, and for some $p \in [0, 1], x_i^t \to p$ for all i as $t \to \infty$.

In other words, the repeated weighted averaging of a CA causes the CA to converge to a fixed point. The proof relies on the fact that averaging values produces a result that is between these two values. Thus, the cell with the smallest value at time t will be averaged with a cell of equal or greater value, and so the minimum value at time $t + 1$ will always be greater or equal to the minimum value at time t. Similarly, the maximum value at time $t + 1$ will always be less than or equal to the maximum value at time t. The proof that the smallest value as $t \to \infty$ is equal to the largest value as $t \to \infty$ is more involved, and the full proof of theorem can be found in [5].

Weighted Average Rules

To show that a FCA is convergent, it is sufficient to show that a FCA is a GWCA for which the general convergence theorem applies. This was shown to be true for several such rules in [5], shown in 4.2.

Period	Rules
Period 1 (Homogeneous)	R_{172}
Period 1 for n even (Heterogeneous)	R_{78}
Period 2 for all n	R_{27}
Period 2 for n even	R_{29}, R_{58}, R_{184}
Period 4 for n multiple of 4	R_{46}

Table 4.2: Observed dynamics of Weighted Average FCA in DNF

The general convergence theorem only applies directly to rule R_{172} (and equivalent rules $R_{212}, R_{202}, R_{228}$. For the other weighted average rules, it is necessary to construct a topologically conjugate system for which the theorem does apply. We omit this, and instead focus on the simplest case of R_{172}, which has the following local transition function:

$$f_{172}(x_{i-1}, x_i, x_{i+1}) = (1 - x_{i-1})x_i + x_{i-1}x_{i+1}$$

So, the state of x_i at time $t + 1$ is the average of x_i and x_{i+1} weighted by x_{i-1} and time t. To apply the general convergence theorem, it is sufficient to show that there exists a value $0 < \gamma < \frac{1}{2}$ such that the weights are bounded by γ and $1 - \gamma$. If $\gamma = minx_0^0, \ldots, x_{n-1}^0, 1 - x_0^0, \ldots, 1 - x_{n-1}^0$ then $\forall t$,

$$\gamma = minx_0^t, \ldots, x_{n-1}^t, 1 - x_0^t, \ldots, 1 - x_{n-1}^t$$
$$1 - \gamma = maxx_0^t, \ldots, x_{n-1}^t, 1 - x_0^t, \ldots, 1 - x_{n-1}^t$$

Thus, $\gamma \leq \gamma_i^t \leq 1-\gamma$ and the convergence theorem hold, thus R_{172} is convergent.

In the process of proving convergence for rules R_{29} and R_{184}, it was discovered that these rules are density conserving in both the boolean and fuzzy domain when its size is even. The sum S of the values of a configuration at time t is conserved at any time $t+2i, \forall i$. Furthermore, the sum of the values at time $t+i, \forall i$, is $n-S$. This means that the density at each iteration of rule R_{29} is completely determined by the initial configuration.

4.1.3 Self-Averaging Rules

Self-oscillating rules belong to a broader class of fuzzy cellular automata called self-averaging rules. Self averaging rules can be written as the weighted average of one of their variables:

$$f(x, y, z) = \gamma(y, z)x + (1 - \gamma(y, z))(1 - x)$$

(analogously for variables y and z). Any fuzzy rule which can be written as

$$f(x_0, \ldots, x_{n-1}) = \gamma(x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1})x_i + (1-\gamma(x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1}))(1-x_i)$$

for some i is a self-averaging rule. In [7], it was proven that all self-averaging rules converge to $\frac{1}{2}$, and are the only rules to do so. Furthermore, all self-averaging rules follow the behaviour of some Boolean rule as the oscillate about $\frac{1}{2}$, as shown in table 4.3. We can see that the self-oscillating rules are a special case of the more general self-averaging rules.

4.2 Relationship between Fuzzy and Boolean CA

When moving from the Boolean domain to a fuzzy logic domain, it invites the question as to what links exist (if any) between these two types of CA. This was first investigated by Mingarelli in [41, 42], who showed that while several Boolean elementary CA appear to be chaotic, none of the rules appeared to be chaotic under DNF fuzzification (or any other fuzzy logic, for that matter). The relationship between

Fuzzy Rule	Equation	Boolean Behaviour
f_{60}	g_{60}	
f_{90}	g_{90}	
f_{105}	g_{105}	
f_{150}	g_{150}	
f_{30}		g_{15}
f_{45}		g_{15}
f_{106}		g_{170}
f_{154}		g_{170}
f_{108}		g_{204}
f_{156}		g_{204}
f_{54}		g_{51}
f_{57}		g_{51}

Table 4.3: Self-averaging rules

Boolean CA and Fuzzy CA later investigated by Betel and Flocchini in [6]. They hoped that showing a link between the two would allow research into the properties of fuzzy CA to become applicable to Boolean CA, and *vice versa*. They succeeded in showing a preservation of the property of density conservation, a relationship between additivity and self-oscillations, and an equivalence between convergent fuzzy CA and the mean field approximation in Boolean CA. Their results are discussed in more detail below.

4.2.1 Density Conservation

Note that all results pertaining to the conservation of the density property are valid only for finite or circular CA, since this property is undefined on an infinite CA.

Temporal Density Conservation

A Boolean CA is said to be *number conserving* if the number of ones in its initial configuration is maintained with each subsequent generation. The analogous

property for fuzzy CA is *sum conservation*, where the sum of all cells is preserved with each generation. This is also called *density conservation*, since the density of a CA is dependent on its sum.

It was discovered that a Boolean CA is density conserving if, and only if, the corresponding FCA is sum preserving.

Spatial Density Conservation

A Boolean CA is said to be *spatially number conserving* if the number of ones in the even numbered cells is equal to the number of ones in the odd numbered cells at any time after the initial configuration. The analogous property for fuzzy CA is *spatial sum conservation*, where the sum of even numbered cells equals the sum of odd numbered cells at any time after the initial configuration. Again, this is also called *spatial density conservation*, as the spatial density depends only on its spatial sum.

It was discovered that Boolean CA is spatially number conserving if, and only if, the corresponding FCA is spatially sum conserving.

4.2.2 Additivity and Self-Oscillation

Additivity

A Boolean CA g is additive if

$$g(y_0, \ldots, y_{n-1}) \oplus g(z_0, \ldots, z_{n-1}) = g(y_0 \oplus z_0, \ldots, y_{n-1} \oplus z_{n-1})$$

In [6], this definition is broadened to include rules that satisfy

$$g(y_0, \ldots, y_{n-1}) \oplus g(z_0, \ldots, z_{n-1}) = \overline{g(y_0 \oplus z_0, \ldots, y_{n-1} \oplus z_{n-1})}$$

When a Boolean CA is additive, it can be expressed as the XOR of some of its variables, and at most one negation.

In fuzzy logic, the NOT operator is defined as $\overline{x} = 1 - x$, and the definition of the XOR operator is extended to the real-valued domain with the definition:

$$\begin{aligned} x \oplus y &= x\overline{y} + \overline{x}y \\ &= x(1-y) + (1-x)y \end{aligned}$$

Using this definition, it can be shown that if a Boolean rule is additive, its fuzzy counterpart is also additive.

Self-Oscillation

Self-oscillation is a property of certain fuzzy CA that converge to a homogeneous fixed point, and deals with the behaviour of values as they oscillate around and approach their fixed point. We begin with a formal definition:

Let G be the global transition function of a Boolean CA, and let F be the global transition function of its fuzzy counterpart. Let p the value of the homogeneous fixed point of F. Let $x_0, \ldots, x_{n-1}$ be some configuration, and let $x_n = F(x_0, \ldots, x_{n-1})$. Now we define y_i as:

$$y_i = \begin{cases} 0 \text{ if } x_i < p \\ 1 \text{ if } x_i > p \end{cases}$$

F is self-oscillating around p if it converges to p, and

$$F(x_0, \ldots, x_{n-1} = x_n \Rightarrow G(y_0, \ldots, y_{n-1}) = y_n$$

.

In other words, self-oscillating fuzzy rules have the property that if you consider values below the convergence point to be zero, and values above the convergence point to be one, then the fuzzy CA will behave like its boolean counterpart as it evolves. Consider, as an example, rule 90:

x	y	z	$f_{90}(x,y,z)$	x	y	z	$g_{90}(x,y,z$
$<$	$<$	$<$	$<$	0	0	0	0
$<$	$<$	$>$	$>$	0	0	1	1
$<$	$>$	$<$	$<$	0	1	0	0
$<$	$>$	$>$	$>$	0	1	1	1
$>$	$<$	$<$	$>$	1	0	0	1
$>$	$<$	$>$	$<$	1	0	1	0
$>$	$>$	$<$	$>$	1	1	0	1
$>$	$>$	$>$	$<$	1	1	1	0

Table 4.4: Fuzzy rule 90 behaviour around $\frac{1}{2}$ (left), Boolean rule 90 (right)

It is clear from table 4.4 that rule 90's oscillation around $\frac{1}{2}$ follows the behaviour of its Boolean counterpart. The same is true for all self-oscillating rules, namely R_60, R_90, R_105, and R_150 (excluding equivalent rules). All self-oscillating rules converge toward $\frac{1}{2}$, the reason for which is described in detail in [6].

Relationship Between Additivity and Self-Oscillation

It was proven in [6] that a Boolean rule is additive if, and only if, its corresponding fuzzy rule is self-oscillating. The proof of this theorem is somewhat complex, and is omitted for brevity.

4.2.3 Convergence and the Mean Field Approximation

The mean field approximation (MFA) is an estimate of the asymptotic density of a Boolean CA, if cells were independent from each other. It is an attempt to predict the number of ones in a Boolean CA after it has stabilized. The MFA is derived by assuming that reaching an asymptotic density implies that the probability of a cell transitioning from 1 to 0 is equal to the probability of a cell transitioning from 0 to 1. Of course, in practise, CA cells are not independent, and correlation between cells renders the mean field approximation inaccurate. Still, the mean field approximation is able to give a rough estimation of asymptotic density in most cases.

One way to interpret fuzzy CA is as a probabilistic version of a Boolean CA. Let x_i^t denote the probability that cell y_i of a Boolean CA assumes value 1 at time t. In this case, the fuzzy rule applied to a neighbourhood returns:

$$f(x_{i-r}^t, \ldots, x_i^t, \ldots, x_{i+r}^t) = x_i^{t+1} = P(y_i^{t+1} = 1)$$

where $P(y_i^t = 1)$ is the probability of a cell having a value of 1 in cell y_i at time t. So, applying the fuzzification of a Boolean CA to the expected values of Boolean cells gives the expected value of the cell at the next time step. The following relationship between fuzzy CA density and the mean field approximation was proven in [6]:

Given a global fuzzy rule F, if, and only if, there exists an homogeneous configuration $X = (p, \ldots, p)$ such that $F(X) = X$, then p is a stable density of the mean field approximation of the Boolean rule G associated with F. In other words, if a fuzzy CA converges to a homogeneous fixed point, then that value is equal to the mean field approximation of its boolean counterpart, and *vice versa*.

4.2.4 Error Propagation

One of the stated goals of Fuzzy CA research is to allow the analysis of Boolean CA with inexact or erroneous data, due to noise, corruption, etc. Indeed, by definition, all fuzzy CA will operate as their Boolean counterparts when given Boolean data. In [7], the propagation of non-Boolean values in an otherwise Boolean CA was studied. Specifically, they examine rules which tend to converge to a fixed-point homogeneous configuration, but cannot converge when given strictly Boolean data. They derive the necessary and sufficient conditions for a single fuzzy value (in the open interval $(0, 1)$) to propagate and "corrupt" the Boolean data (in the set $\{0, 1\}$) and allow the values to converge. We present several theorems which were proven in [7], below.

- A single non-binary value in the initial configuration is necessary and sufficient to force rules 105, 150, 54, 57 to converge.
- For rule 90, two fuzzy values are required for convergence; one in an odd-numbered cell, and one in an even-numbered cell.
- For rule 60, one fuzzy value will propagate to the right, but not the left. (For a circular CA, the fuzzy value will wrap around and cause total convergence.)
- Given a single fuzzy value in its initial configuration, rule 108 will converge to one half if, and only if, the binary data to the right of the fuzzy value has a

repeating pattern of the form $(1110)^\infty$, $(1101)^\infty$, $(1011)^\infty$, $(0111)^\infty$, or $0(1011)^\infty$. To the left of the fuzzy value, the data must be a mirror image of one of these configurations.

- Given a single fuzzy value in the initial configuration of rule 156, all values will eventually converge toward $\frac{1}{2}$ if, and only if, the initial configuration is of the form $\cdots 11111111axb11111111\cdots$, where $a, b \in \{0, 1\}$, and x is the fuzzy value.
- Rule 45 will converge to $\frac{1}{2}$ if, and only if, the data to the left of the initial fuzzy value does not contain an infinite sequence of $(001)^\infty$.
- Rule 30 will converge to $\frac{1}{2}$ if, and only if, the data to the left of the fuzzy value does not contain the left infinite sequence $\cdots, 0, 1, 0, 1, 0, 1$.
- Rule 154 will converge to $\frac{1}{2}$ if, and only if, there is an even number of 0s before the first 1 to the left of the fuzzy data and the data to the right of the fuzzy data does not contain an infinite sequence of 1s.
- If in the initial configuration the fuzzy data is not both preceded and followed by a 0, then rule 106 will eventually converge to one half, if, and only if the data to the right of the initial fuzzy value does not contain an infinite sequence of 0s or 1s.

4.3 Applications of Fuzzy CA

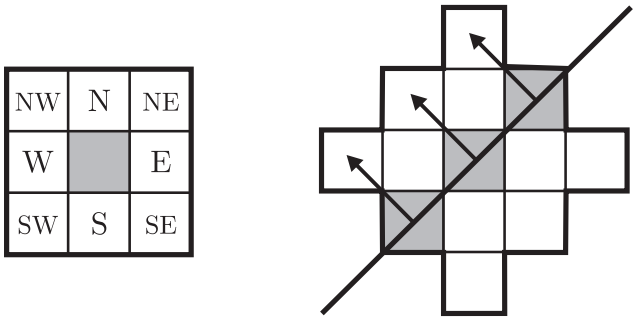
4.3.1 Image Sharpening and Edge Detection

A bitmap image is the most common format for computers to store and process graphical data. It consists of an two-dimensional array of colour values, called pixels.

This is essentially the same format as the configuration of a two-dimensional CA. By applying a suitable local transition function, the image will be transformed in some way from one generation to the next. In this way, FCA can be used to perform image processing.

In [39], a method for image sharpening using FCA is introduced. The local transition function consists of an algorithm that calculates the likelihood that an edge exists at a given cell. An edge in a bitmap image refers to a region of the image that has a strong change in value (brightness) when traversing pixels in a certain direction. It then uses that result to apply sharpening algorithm that is applied locally to the given cell.

The *simple derivative* at cell (x, y) is direction D is defined as the difference between the value of the cell and its neighbour in direction D. A direction is one of eight basic compass directions: NW, W, SW, S, SE, E, NE, and N. See fig. 4.1. The simple derivative is denoted $\nabla_D(x, y)$. For example, $\nabla_N(x, y) = I(x, y - 1) - I(x, y)$. (They use standard Cartesian co-ordinates, but with an inverted y-axis.)



Neighbourhood of pixel Edge passing through pixel

Figure 4.1: Neighbourhoods used in calculating fuzzy derivative

The *fuzzy derivative* extends the concept of the simple derivative to by averaging the derivatives of neighbouring cells. Consider the edge passing diagonally through figure 4.1. Not only will the central cell have a large value for $\nabla_N W$, but the NE and

SW cells will also have a large value for $\nabla_N W$. If at least two of the three cells have a large value for $\nabla_N W$, then it is safe to assume that an edge is passing diagonally through the cell. The choice of the threshold for what is a "large" simple derivative is a parameter that can be varied to achieve qualitatively different results.

If the cell is determined to belong to an edge, a sharpening algorithm is then applied to the value of the cell, which is beyond the scope of this discussion. A sample of this FCA applied to an image is shown below in figure 4.2.

Original image Sharpened image

Figure 4.2: FCA Image sharpening

In [60], this method was applied to improve traditional methods of edge detection of noisy images. Noise is a particular problem for edge detection algorithms, because noise is often erroneously detected as an edge. At each iteration, cells that are likely edges are brightened, and cells that are likely not edges are darkened. This process is repeated iteratively until the FCA evolves into a steady state. An example of this process is shown below:

In figure 4.3, we see that this method produces similar results to traditional edge detection algorithms.

In figure 4.4, we see that the FCA method produces superior results to traditional edge detection algorithms when processing images with a lot of noise.

Original Image Traditional Edge Detection FCA Edge Detection

Figure 4.3: Traditional edge detection vs. FCA

Noisy Source Traditional Edge Detection FCA Edge Detection

Figure 4.4: Traditional edge detection vs. FCA with a noisy source

4.3.2 Fuzzy Hexagonal CA and Snowflakes

In [57], Stephen Wolfram demonstrates that two-dimensional hexagonal CA can be used to generate images that mimic the growth of snowflake crystals. One can visualise a hexagonal CA as a hexagonal tessellation of the plane. Cells in a hexagonal CA have exactly six immediate neighbours. (fig. 4.5) In Wolfram's snowflakes, a value of one is known as *frozen*, while a value of zero is known as *not frozen*.

In order for a CA to evolve symmetrically, the initial conditions must be symmetrical. To create a symmetrical snowflake, all cells are set to zero, except for one cell in the centre which is set to a value of one. This cell is known as the *seed*. The local transition function is as follows: if a cell is frozen, it stays frozen. If a cell is not frozen, it will become frozen if it has exactly one frozen neighbour. In the presence of the seed cell, the CA rule will cause adjacent cells to become frozen as well. The combination of symmetrically initial conditions, and a symmetrical rule, causes the

snowflake to appear to *grow* symmetrically from the seed. An example evolution is shown in figure 4.8.

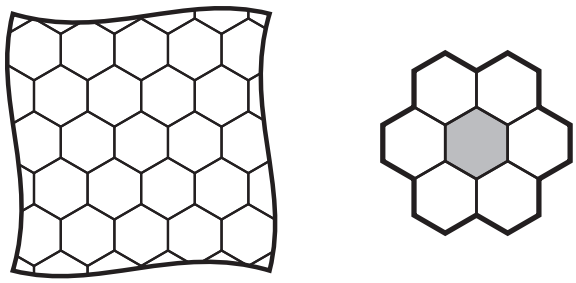


Figure 4.5: Hexagonal tessellation and neighbourhood

Wolfram's simplistic snowflake cannot be altered without destroying its behaviour, and thus does not exhibit the diversity of growth patterns observed in nature. Coxe investigated fuzzy hexagonal CA to improve the behaviour of snowflake growth in [15]. The goal of this investigation was to create a CA rule that exhibited the same symmetrical growth as Wolfram's snowflakes, yet exhibited a great deal of sensitivity to initial conditions.

In this case, the value of each cell represents the probability that that cell is frozen. Cells with a value greater than or equal to 0.5 are considered to be frozen, while cells less than 0.5 are not frozen. As with Wolfram's snowflake, all cells are set to a constant value which is less than 0.5, except for one cell in the centre which is set to a value of 1.0. This cell is known as the *seed*. The neighbourhood is increased to a radius of two (see fig. 4.6).

Each neighbourhood is divided into four types. The central cell (C), immediate neighbours (N), points (P), and edges (E). The location transition function is given by:

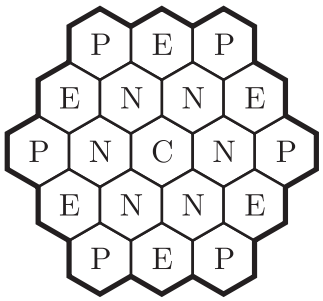


Figure 4.6: Hexagonal snowflake neighbourhood

$$f(x) = \alpha C + \beta \sum N + \gamma \sum P + \delta \sum E$$

α, β, γ, and δ vary depending on the number of frozen cells of each type that exist. The choices for these parameters are arbitrary, and can be varied to produce snowflakes with different appearances. Coxe showed that even tiny variations in these parameters can produce vastly different results.

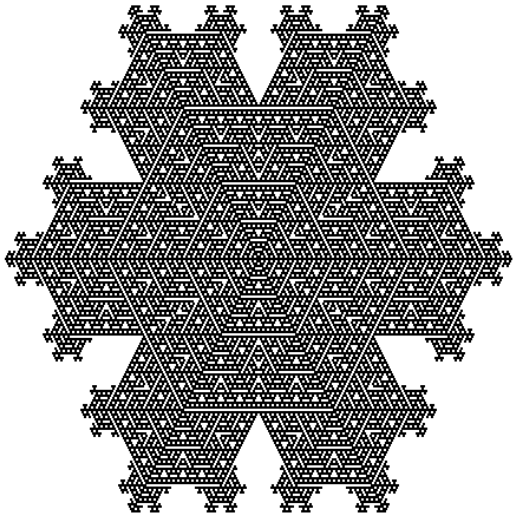


Figure 4.7: Wolfram's Boolean hexagonal snowflake

Note that both Wolfram's and Coxe's snowflakes demonstrate an interesting property of cellular automata. A CA with symmetrical initial conditions, and a sym-

Figure 4.8: Coxe's fuzzy hexagonal snowflake

metrical local transition function, will evolve symmetrically. This can cause identical complex structures to appear in different locations of the CA simultaneously, despite the fact that these cells cannot observe each other directly. This highlights the fact that even seemingly random CA are actually deterministic.

4.4 Conclusion

In this chapter, we have reviewed how FCA in DNF have been classified, both empirically and analytically. We began by introducing the initial empirical classification. We then introduced the concept of weighted average rules and self-averaging rules as a means of understanding the behaviour of FCA in DNF. We then discussed the relationship between FCA and Boolean CA, in terms of density conservation, and the relationship between additivity and self-oscillation. Finally, we explored a few applications of FCA in DNF.

It is important to note once again that, despite expectations, no FCA in DNF

exhibits chaos, but instead they all appear to be asymptotically periodic. This will become relevant in the next chapter, in which we prove that FCA in CNF do, in fact, exhibit chaos.

It is also worth noting that the initial classification of FCA in DNF was performed empirically, and only later analytically. The empirical simulations exhibited behaviour that appeared complex and interesting, but were not completely understood initially. In the next chapter, we perform a simulation of FCA in CNF, and perform a empirical classification similar to the empirical classification in DNF. While we do not succeed in completely explaining the behaviour FCA in CNF in general, we do perform a some similar analysis of FCA in CNF in certain exceptional cases.

Chapter 5

Fuzzy CA in Conjunctive Normal Form with CFMS Logic

In the previous chapter, we have seen how fuzzy CA in the disjunctive normal form can be classified based on the observed asymptotic behaviour of the CA evolved from a random initial configuration [10, 20, 5]. While the FCA in the disjunctive normal form has been fairly well classified, the same is not the case for FCA in the conjunctive normal form. For this reason, we wish to study CNF FCA.

We began our study by analyzing the asymptotic behaviour of FCA in CNF with CFMS logic algebraically. Although we were unable to do so in the general case due to its complexity, we did succeed in solving these equations in the special case of a homogeneous initial configuration. We propose four classifications for the homogeneous case, and prove that one of these classes is chaotic, in the classical sense. Note that our results in the homogeneous case are applicable to CA of both finite (circular) and infinite size.

We then created computer simulations of all 256 elementary CA rules in CNF

with CFMS logic with random initial configurations, observing their asymptotic behaviour. We propose 7 classifications based on our empirical observations. When possible, we try to explain these observed behaviours. Note that our results in this case are valid only for finite CA with circular boundary conditions, as this is a limitation of our simulation.

5.1 Naming Convention and Definitions

5.1.1 Naming Convention

We propose the following naming convention for our classes:

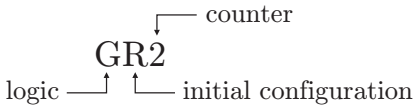


Figure 5.1: Naming Convention

Each class is given a 3 character code, as shown in figure 5.1. The first character specifies the fuzzification logic, as per table 5.1. The second character specifies whether a homogeneous or random initial configuration was used, as per table 5.2. The final character is an arbitrary number, generally assigned in the order in which each class was found. The class shown in figure 5.1, for example, is G¨odel logic with a random initial configuration, and it is the second class of that type.

5.1.2 Definitions

Recall the definition of CFMS fuzzification:

As an exercise, consider rule $183 = 128 + 32 + 16 + 4 + 2 + 1$:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (1, 0, 1, 1, 0, 1, 1, 1)$$

The conjunctive normal form expression of rule 183 is:

$$f_{183}(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$$

The fuzzification process yields:

$$min\{1, (-x_1 - x_2 + x_3 + 2)\} \cdot min\{1, (x_1 - x_2 - x_3 + 2)\}$$

This fuzzification process can be quite cumbersome. To simplify the process, table 5.4 may be used in the construction of the local transition function of an elementary CNF FCA using CFMS logic.

Since the conjunction operator is represented as multiplication in CFMS, the presence of a 0 in a particular column of the rule's binary representation indicates the presence of the given factor in its transition function. (Columns are counted from the right, starting with column 0.) Rule 183 has the binary number $(1, 0, 1, 1, 0, 1, 1, 1)$, which has two zeros: in column 3 and column 6. So, the CFMS fuzzification of Rule

Logic	Code
CFMS	C
Probabilistic	B
Łukasiewicz	L
Gödel	G
Zadeh	Z
Product	D

Table 5.1: Logic Codes

Initial Configuration	Code
Homogeneous	H
Random	R

Table 5.2: Initial Configuration Codes

Boolean	Fuzzy
$\neg x$	$1 - x$
$x \wedge y$	$x \cdot y$
$x \vee y$	$min\{1, x + y\}$

Table 5.3: CFMS Fuzzification

Column	Value	Transition	Factor
0	1	$(0,0,0) \to 0$	$min\{1, (x_1 + x_2 + x_3)\}$
1	2	$(0,0,1) \to 0$	$min\{1, (x_1 + x_2 - x_3 + 1)\}$
2	4	$(0,1,0) \to 0$	$min\{1, (x_1 - x_2 + x_3 + 1)\}$
3	8	$(0,1,1) \to 0$	$min\{1, (x_1 - x_2 - x_3 + 2)$
4	16	$(1,0,0) \to 0$	$min\{1, (-x_1 + x_2 + x_3 + 1)\}$
5	32	$(1,0,1) \to 0$	$min\{1, (-x_1 + x_2 - x_3 + 2)\}$
6	64	$(1,1,0) \to 0$	$min\{1, (-x_1 - x_2 + x_3 + 2)\}$
7	128	$(1,1,1) \to 0$	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$

Table 5.4: FCA CFMS CNF Factors

183 in CNF is simply the product of the two factors indicated in table 5.4 for those columns.

5.2 Classification with Homogeneous Initial Configuration

During our initial investigation, it became clear to us that analyzing FCA in CNF would prove more difficult than the analysis of FCA in DNF. This is due to the fact that DNF, when "fuzzified", produces a sum of products, while CNF, when "fuzzified", produces a product of sums. Due to the distributive law, simplifying a product of sums can quickly produce a large number of terms. Consider, for example,

the local transition function of FCA CNF rule 0:

$$\begin{aligned} f(a,b,c) = (min(((1-a)+(1-b)+(1-c)),1)) \cdot (min(((1-a)+(1-b)+(c)),1)) \\ \cdot (min(((1-a)+(b)+(1-c)),1)) \cdot (min(((1-a)+(b)+(c)),1)) \\ \cdot (min(((a)+(1-b)+(1-c)),1)) \cdot (min(((a)+(1-b)+(c)),1)) \\ \cdot (min(((a)+(b)+(1-c)),1)) \cdot (min(((a)+(b)+(c)),1)) \end{aligned} \tag{5.1}$$

Clearly, this becomes difficult to simplify and analyze. In order to investigate further, we restricted our analysis to the exception case of having a homogeneous initial configuration, where all cells are assigned the same value. In this case, each cell has the same value, which is identical to its neighbours' values, allowing us to simplify the local transition function. Consider, for example, the local transition function of FCA CNF rule 0 with a homogeneous configuration, after simplification:

$$f(x,x,x) = min\{1,(3-3x)\} \cdot min\{1,(3x)\}$$

For this reason, we are able to study FCA in CNF with a homogeneous configuration in more detail. Recall from table 5.4 that the local transition function $f(x_1,x_2,x_3)$ of a FCA in CNF is a product of factors whose presence (or absence) is determined by the presence (or absence) of transitions that go to 0. In the case of a homogeneous configuration, $x_1 = x_2 = x_3 = x$, so we can simplify the products from table 5.4. We can further simply the factors by applying the knowledge that the value of any cell in the CA is in the range $[0,1]$. (We impose this restriction on the initial configuration, and we know this property is preserved by our choice of transition func-

tion.) The following table simplifies the factors from table 5.4 in the homogeneous case:

Transition	**CNF Factors**	**Factors When** $\boldsymbol{x_1 = x_2 = x_3 = x}$	**Factors When** $\mathbf{0 \leq x \leq 1}$
$(0,0,0) \to 0$	$min\{1, (x_1 + x_2 + x_3)\}$	$min\{1, (3x)\}$	$min\{1, (3x)\}$
$(0,0,1) \to 0$	$min\{1, (x_1 + x_2 - x_3 + 1)\}$	$min\{1, (1 + x)\}$	1
$(0,1,0) \to 0$	$min\{1, (x_1 - x_2 + x_3 + 1)\}$	$min\{1, (1 + x)\}$	1
$(0,1,1) \to 0$	$min\{1, (x_1 - x_2 - x_3 + 2)\}$	$min\{1, (2 - x)\}$	1
$(1,0,0) \to 0$	$min\{1, (-x_1 + x_2 + x_3 + 1)\}$	$min\{1, (1 + x)\}$	1
$(1,0,1) \to 0$	$min\{1, (-x_1 + x_2 - x_3 + 2)\}$	$min\{1, (2 - x)\}$	1
$(1,1,0) \to 0$	$min\{1, (-x_1 - x_2 + x_3 + 2)\}$	$min\{1, (2 - x)\}$	1
$(1,1,1) \to 0$	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$	$min\{1, (3 - 3x)\}$	$min\{1, (3 - 3x)\}$

Table 5.5: Homogeneous CFMS FCA CNF Factors

Recall that CA functions in CNF are the product of factors whose presence (or absence) is determined by the presence (or absence) of transitions that go to 0. From table 5.5 we see that in the homogeneous case, most of these factors can be simplified to 1, regardless of the input values. The presence (or absence) of factors of 1 do not affect the resultant value of a function of products. We can therefore safely ignore factors that go to 1. Therefore, we can say that in the homogeneous case, the local transition rule of CFMS FCA in CNF can be determined solely by the presence (or absence) of the following two transitions: $(1,1,1) \to 0$, and $(0,0,0) \to 0$. Furthermore, since there are only four combinations of the above two transitions being present or not, we can divide all homogeneous rules canonically into four classes.

We define the following four classes of homogeneous CFMS CNF FCA, along with their local transition function (note: The binary representation of a class of rules includes a third digit, $*$, where $* \in \{0, 1\}$, known as a "don't care" state):

We discuss each homogeneous class below:

Class	Binary	Rules r_z	$f(x,x,x)$
CH1	1******1	$z \in [128, 256), z \bmod 2 \neq 0$	1
CH2	1******0	$z \in [128, 256), z \bmod 2 = 0$	$min\{1, (3x)\}$
CH3	0******1	$z \in [0, 128), z \bmod 2 \neq 0$	$min\{1, (3-3x)\}$
CH4	0******0	$z \in [0, 128), z \bmod 2 = 0$	$min\{1, (3-3x)\} \cdot min\{1, (3x)\}$

Table 5.6: Homogeneous CFMS FCA CNF Classes

5.2.1 CFMS Homogeneous Class 1 (CH1)

Class CH1 rules include all rules with the transitions $(0,0,0) \to 1$ and $(1,1,1) \to 1$, that is all odd rules greater than or equal to 128. Class CH1 rules have the local transition function of

$$f(x,x,x) = 1$$

and converge immediately to the fixed point configuration of $X^t = (1)^n$, $t > 0$ in one step. From this we can make the following statement:

- $\forall \tau > 0, X^\tau = (1)^n$

5.2.2 CFMS Homogeneous Class 2 (CH2)

Class CH2 rules include all rules with the transitions $(0,0,0) \to 0$ and $(1,1,1) \to 1$, that is, all even rules greater than or equal to 128. Class CH2 rules have the local transition function of $f(x,x,x) = min\{1, (3x)\}$. By solving the inequality $3x \leq 1$, we can eliminate the min function by splitting Class B into the following two cases:

$$f(x,x,x) = \begin{cases} 3x, & 0 \leq x \leq \frac{1}{3} \\ 1, & \frac{1}{3} < x \leq 1 \end{cases}$$

By solving for x in $f(x,x,x) = x$ in both of these cases, we can easily see that

Class CH2 rules have two fixed point configurations: $X = (0)^n$ and $X = (1)^n$. We can also say that $f(x, x, x)$ is a non-decreasing function; it is fixed at $x \in \{0, 1\}$, and it is strictly increasing when $x \in (0, 1)$. From this we can easily see that any rule of class CH2 converges to $(1)^n$ from any initial configuration except from $X^0 = (0)^n$, in which case the rules converge to $(1)^n$. So, we can state:

Theorem 5.1: If $X^0 = (0)^n$, then $\forall \tau > 0, X^\tau = (0)^n$

If $X^0 = (x)^n$ where $x \neq 0$ then $\exists \tau > 0, \forall m \geq 0, X^{\tau+m} = (1)^n$

5.2.3 CFMS Homogeneous Class 3 (CH3)

Class CH3 rules include all rules with the transitions $(0, 0, 0) \rightarrow 1$ and $(1, 1, 1) \rightarrow 0$, that is all odd rules smaller than 128. Class CH3 rules have the local transition function of $f(x, x, x) = min\{1, (3 - 3x)\}$. By solving the inequality $3 - 3x \leq 1$, we can eliminate the min function by splitting Class CH3 into the following two cases:

$$f(x, x, x) = \begin{cases} 1, & 0 \leq x < \frac{2}{3} \\ 3 - 3x, & \frac{2}{3} \leq x \leq 1 \end{cases}$$

We can show that any rule of class CH3 has a unique fixed point $X = (\frac{3}{4})^n$. Moreover, for any initial configuration $X^0 = (x)^n$ with $x \neq^3 /_4$ the rule converges toward a periodic behaviour cycling between $(0)^n$ and $(1)^n$.

Theorem 5.2: If $X^0 = (\frac{3}{4})^n$, then $\forall \tau > 0, X^\tau = (\frac{3}{4})^n$

If $X^0 = (x)^n$ where $x \neq \frac{3}{4}$ then $\exists \tau > 0, \forall m \geq 0, X^{\tau+2m} = (1)^n$ and $X^{\tau+2m+1} = (0)^n$

Proof. By solving for x in $f(x, x, x) = x$, we can easily see that Class CH3 rules

have one fixed point configuration: $X = (\frac{3}{4})^n$. We can also see the following periodic behaviour: If $X^t = (0)^n$, then $X^{t+1} = (1)^n$, and if $X^t = (1)^n$, then $X^{t+1} = (0)^n$.

We now prove that, for any other $x \neq \frac{3}{4}$, the function will diverge from this value until it enters the interval $[0, \frac{2}{3}]$ or becomes equal to 1 thus starting the periodic behaviour. Let d be any positive value smaller than or equal to $\frac{1}{4}$, and let $x^t = \frac{3}{4} + d$.

$$\begin{aligned} f(x, x, x) &= 3 - 3(\frac{3}{4} + d) \\ &= \frac{3}{4} - 3d \end{aligned}$$

So, applying f we have that x^{t+1} produces a value that is smaller than $\frac{3}{4}$ and further apart from it. More precisely $|x^{t+1} - \frac{3}{4}| = 3|x^t - \frac{3}{4}|$. This behaviour will cause the value to oscillate around and diverge from $\frac{3}{4}$ until either $x < \frac{2}{3}$ or $x \geq 1$. At that point, $f(x, x, x)$ will fall into the periodic behaviour mentioned above. □

5.2.4 CFMS Homogeneous Class 4 (CH4)

Class CH4 rules include all rules with the transitions $(0, 0, 0) \rightarrow 0$ and $(1, 1, 1) \rightarrow 0$, that is, all even rules smaller than 128. Class CH4 rules have the local transition function of $f(x, x, x) = min\{1, (3 - 3x)\} \cdot min\{1, (3x)\}$. It is a product of Class CH2 and Class CH3. We can eliminate the min function by splitting Class CH4 into the following cases:

$$f(x, x, x) = \begin{cases} 3x, & 0 \leq x \leq \frac{1}{3} \\ 1, & \frac{1}{3} \leq x \leq \frac{2}{3} \\ 3 - 3x, & \frac{2}{3} \leq x \leq 1 \end{cases}$$

We now study the properties of this class of rules. We first determine if the above function has any fixed point configurations. By solving for x in $f(x,x,x) = x$, we have the following fixed points:

$\boldsymbol{x}$	$\boldsymbol{f(x,x,x)}$	$\boldsymbol{f(x,x,x) = x}$
$0 \leq x \leq \frac{1}{3}$	$3x$	$x = 0$
$\frac{2}{3} \leq x \leq 1$	$3 - 3x$	$x = \frac{3}{4}$

Table 5.7: Class CH4 Fixed Points

So, we can see that:

Theorem 5.3: Class CH4 rules have two fixed point configurations: $X = (0)^n$, and $X = (\frac{3}{4})^n$.

We now consider the periodic points of $f(x,x,x)$, starting with periods of length 2. Since $f(x,x,x)$ is discontinuous, solving for periodic configurations becomes more difficult. For brevity, we ignore cases where $f[f(x,x,x)] = 1$, since this is known not to be periodic.

Theorem 5.4: Class CH4 rules have only one homogeneous configuration with period 2

Proof. By direct calculation (see Table 5.8), we can produce all periodic configurations of period 2, which include fixed point configurations. By eliminating fixed point configurations, and verifying that the values satisfy $f[f(x,x,x)] = x$, we can see that there is only one pair of homogeneous configurations with period 2: $(0.3)^n$ and $(0.9)^n$ (i.e., $F((0.3)^n) = (0.9)^n$ and $F((0.9)^n) = (0.3)^n$.

□

We can also deduce some values that will eventually converge to this periodic

$\boldsymbol{x}$	$\boldsymbol{f(x,x,x)}$	$\boldsymbol{f(f(x,x,x))}$	$\boldsymbol{f(f(x,x,x))=x}$
$0 \leq x \leq \frac{1}{9}$	$3x$	$9x$	$x=0$
$\frac{2}{9} \leq x \leq \frac{3}{9}$	$3x$	$9-9x$	$x=\frac{9}{10}$
$\frac{6}{9} \leq x \leq \frac{7}{9}$	$3-3x$	$9x-6$	$x=\frac{3}{4}$
$\frac{8}{9} \leq x \leq 1$	$3-3x$	$3-9x$	$x=\frac{3}{10}$

Table 5.8: Class CH4 Period 2

configuration. For example, it is easy to show that any value of the form $x=\frac{0.3}{3^k}$, where $k \geq 0$ will be transformed into 0.3 after undergoing the function $f^k(x,x,x)=3x$.

Theorem 5.5: Any initial configuration of the form $X^0=(x)^n$ with $x=\frac{0.3}{3^k}$ for any $k \geq 0$ converges to the periodic configuration $(0.3)^n,(0.9)^n \ldots^n$.

Proof. Recall from Class CH3 that $f(\frac{3}{4}+d)=\frac{3}{4}-3d$. By setting $\frac{3}{4}-3d=0.9$ and solving for d, we have $d=\frac{0.15}{-3}=-0.05$, so $x=\frac{3}{4}+(-0.05)=0.7$. $f(0.7)=0.9$, which produces the same periodic behaviour. By applying this method recursively, we can generalize this to say that $x=\frac{3}{4}+\frac{0.15}{(-3)^k}$, where $k \geq 1$ will be transformed into 0.9 after undergoing the function $f^k(x,x,x)=3-3x$. □

We now determine if the above function has any periodic configurations with period 3 and we obtain:

Theorem 5.6: All periodic configurations of period 3 are the following:

- $f((\frac{3}{28})^n)=(\frac{9}{28})^n$, $f((\frac{9}{28})^n)=(\frac{27}{28})^n$, and $f((\frac{27}{28})^n)=(\frac{3}{28})^n$
- $f((\frac{6}{26})^n)=(\frac{18}{26})^n$, $f((\frac{18}{26})^n)=(\frac{24}{26})^n$, and $f((\frac{24}{26})^n)=(\frac{6}{26})^n$

Proof. By direct calculation of $f(f(f(x,x,x)))$ we obtain:

Note that this procedure produces all periodic configurations of period 3, which includes fixed point configurations. By eliminating fixed point configurations, and

$\boldsymbol{x}$	$\boldsymbol{f(x,x,x)}$	$\boldsymbol{f(f(x,x,x))}$	$\boldsymbol{f\{f(f(x,x,x))\}}$	$\boldsymbol{f\{f(f(x,x,x))\}=x}$
$0 \leq x \leq \frac{1}{27}$	$3x$	$9x$	$27x$	$x = 0$
$\frac{2}{27} \leq x \leq \frac{3}{27}$	$3x$	$9x$	$3 - 27x$	$x = {}^{3}/_{28}$
$\frac{6}{27} \leq x \leq \frac{7}{27}$	$3x$	$3 - 9x$	$27x - 6$	$x = {}^{3}/_{13}$
$\frac{8}{27} \leq x \leq \frac{9}{27}$	$3x$	$3 - 9x$	$9 - 27x$	$x = {}^{9}/_{28}$
$\frac{18}{27} \leq x \leq \frac{19}{27}$	$3 - 3x$	$9x - 6$	$27x - 18$	$x = {}^{9}/_{13}$
$\frac{20}{27} \leq x \leq \frac{21}{27}$	$3 - 3x$	$9x - 6$	$21 - 27x$	$x = {}^{3}/_{4}$
$\frac{24}{27} \leq x \leq \frac{25}{27}$	$3 - 3x$	$9 - 9x$	$27x - 24$	$x = {}^{12}/_{13}$
$\frac{26}{27} \leq x \leq 1$	$3 - 3x$	$9 - 9x$	$27 - 27x$	$x = {}^{27}/_{28}$

Table 5.9: Class CH4 Period 3

verifying that the values satisfy $f\{f(f(x,x,x))\} = x$, we can see that there are the two sets of values of period 3 stated in the theorem. □

Clearly, it would be difficult to determine which initial homogeneous configurations will eventually converge to a periodic configuration. For example, configurations of the form $(\frac{0.3}{3^k})^n$, with $k \geq 0$ converge to a temporally periodic configuration, but there are an infinite number of such configurations, and that is but one example. To make this task easier, we will begin with the set $(0, 1)$, and subtract the complement of this set, namely, the set of all values that converge to the fixed point $(0)^n$.

We begin with the open interval $(0, 1)$ (figure 5.2).

Figure 5.2: Step 0

By definition, for the values of $\frac{1}{3} \leq x \leq \frac{2}{3}$, $f(x) = 1$. We will therefore delete these values from our set, as we know they will not produce a periodic configuration (figure 5.3):

Now we can compute the range of values x such that $\frac{1}{3} \leq f(x,x,x) \leq \frac{2}{3}$. When we delete these values from our set, we have:

Figure 5.3: Step 1

Figure 5.4: Step 2

Now we can compute the range of values x such that $\frac{1}{3} \leq f^2(x,x,x) \leq \frac{2}{3}$. When we delete these values from our set, we have:

Figure 5.5: Step 3

Clearly, a pattern has emerged from this recursive operation. The set of values that do not converge to $X = (0)^n$ is known as the *open Cantor set* (OCS).

Theorem 5.7: Any initial configuration $X^0 = (x)^n$ with $x \in OCS$ does not converge to $X = (0)^n$

5.2.5 Chaotic Behaviour of Class CH4

We have seen that class CH4 rules have interesting and complex behaviour, but do they exhibit chaos? To more easily answer that question, we will map the real numbered space we have been using to another, homeomorphic space, called the ternary sequence space. We will then define a function on this space that is conjugate to our class CH4 function on real numbers. Once in this form, we can then demonstrate that class CH4 rules are, in fact, chaotic.

Ternary Expansion

We begin by mapping our real-numbered values from the decimal (base 10) number system to the ternary (base 3) number system. The *ternary* number system has only three digits: 0, 1, and 2. Knowing that our function $f(x)$ is only defined for the domain $0 \leq x \leq 1$, a sequence of integers $0.a_1a_2a_3\ldots$, where a_i is either 0, 1, or 2 is called the ternary expansion of x if

$$x = \sum_{i=1}^{\infty} \frac{a_i}{3^i}$$

We have seen that the values belonging to a periodic trajectory of the class CH4 function belong to the open Cantor set. The open Cantor set divides the interval of $(0, 1)$ into thirds, which are then subdivided into thirds, *et cetera*. The fractional part of a ternary number also divides the interval of $(0, 1)$ into thirds. If x has ternary expansion $0.a_1a_2a_2\ldots$, the digit a_1 determines to which third of the interval $(0, 1)$ x lies. I.e., if $a_1 = 0, 1$, or 2, then x belongs to the left, middle, or right interval, respectively. The second digit, a_2 indicates in which third of that subinterval x lies. As with all place-value notations, this subdivision is repeated recursively until it either terminates, repeats, or continues infinitely without repetition (in the case of irrational numbers).

Now that our values are in ternary notation, we introduce some properties specific to values belonging to the open Cantor set, which also apply to values belonging to a periodic trajectory of the class CH4 function. First, we know that the open Cantor set does not contain the middle third of every subinterval. Therefore, the ternary expansion of any value belonging to the open Cantor set will not contain the digit 1.

Property 5.1: The open Cantor set contains all real numbers in $(0, 1)$ for which any

ternary expansion contains no 1s.

However, there is a caveat in this property. A real number could have different ternary expansions. For example, $\frac{1}{3}$ has expansion $0.10000\ldots$, but also $0.02222\ldots$. We need to be certain that the values we have excluded in property 5.1 do not have equivalent forms that we have failed to exclude. We also need to be concerned with values belonging to the extremes of each subinterval; In contrast to the standard Cantor set, the *open* Cantor set *does not* contain values belonging to the extremes of each subinterval (e.g. $\frac{1}{3}$, $\frac{2}{9}$, $\frac{19}{27}$). Fortunately, both of these issues can be resolved by observing that:

Property 5.2: All rational numbers of the form $\frac{p}{3^k}$, for some integer $0 \leq p \leq 3^k$, have more than one ternary expansion, while all other numbers have a unique one.

Note that numbers of the form $\frac{p}{3^k}$ are precisely the ones we wish to exclude for belonging to the extremes of each subinterval, and they also happen to be the only numbers with multiple ternary expansions. A number of the form $\frac{p}{3^k}, 0 \leq p \leq 3^k$ can always be written as the ternary expansion $0.a_1 \ldots a_n 2222\ldots$ or $0.a_1 \ldots a_n 0000\ldots$. It then follows that:

Property 5.3: A ternary expansion that does not contain any 1s has an equivalent ternary expansion containing some 1s if and only if it is of the form: $0.a_1 \ldots a_n 022222\ldots$ or $0.a_1 \ldots a_n 20000000\ldots$

So, from all of this, we have proven the following:

Theorem 5.8: The open Cantor set K' contains all real numbers in $0, 1$ whose ternary expansion contains no 1s and does not terminate with infinite 0s or infinite 2s.

Using this knowledge, we can demonstrate one final property of the class CH4

function that we will need later. We can show that if we apply the class CH4 function f to a real number in K', the result is still in K'.

Lemma 5.1: Let $x \in K'$. Then we have: $f(x) \in K'$.

Proof. Let $0.a_1a_2a_3\ldots$ be a ternary expansion of $x \in K'$. We now divide our proof into two exhaustive cases:

Case 1: If $0 < x < \frac{1}{3}$, then $a_1 = 0$, and $f(x) = 3x$. The ternary expansion of $3x$ is then $0.a_2a_3a_4\ldots$, which is still in K'.

Case 2: If $\frac{2}{3} < x < 1$, then $a_1 = 2$, and $f(x) = 3(1-x)$. The ternary expansion of $3(1-x)$ is then $0.\bar{a_2}\bar{a_3}\bar{a_4}\ldots$, where $\bar{a_i} = 2$ when $a_i = 0$, and $\bar{a_i} = 0$ when $a_i = 2$. This ternary expansion is also still in K'.

So, in both cases, $f(x) \in K'$. □

Binary Sequence Space (Symbolic Dynamics)

A *sequence* space is the set of all possible sequences of values belonging to a particular set. These sequences may be either finite in length, infinite, or bi-infinite in length. Sequence spaces may be constructed on any type of set, such as real numbers, complex numbers, or integers. We will only consider infinite sequences on a finite set of symbols. Specifically, we are interested in the sequence space on two symbols, called the *binary sequence space*, which we have denoted Σ_2:

$$\Sigma_2 = \{(s_0s_1s_2\ldots)|s_j \in \{0,1\}\}$$

We convert the ternary expansion of a value K' in the interval $(0,1)$ to a binary

sequence using the itinerary function, $S : K' \to \Sigma_2$:

$$S(0.a_1a_2a_3\ldots) = b_1b_2b_3$$

$$\text{where } b_i = \begin{cases} 0, & \text{if } a_i = 0 \\ 1, & \text{if } a_i = 2 \end{cases}$$

Note that we do not have to consider the case where $a_i = 1$, since we have excluded these points from consideration in lemma 5.1. Also note that in theorem 5.8, we established that values in the open Cantor set do not terminate with infinite 0s or infinite 2s. So, the function S will map values in K' to a subset of Σ_2 that does not terminate with infinite 0s or infinite 1s. We will call this new subset the *modified sequence space*, denoted Σ_2', so that $S : K' \to \Sigma_2'$.

Shift Map

The shift map $\sigma : \Sigma \to \Sigma$ is a function defined by:

$$\sigma(s_0s_1s_2\ldots) = s_1s_2s_3\ldots$$

So, σ removes the first element from the sequence, which shifts everything to the left. We need to define a function on this sequence space that is conjugate to our class CH4 function on real numbers. Recall that the class CH4 function, $f(x), x \in K'$, is defined as

$$f(x)\begin{cases} 3x, & 0 \leq x \leq \frac{1}{3} \\ 3(1-x), & \frac{2}{3} \leq x \leq 1 \end{cases}$$

So, in each case, we need to multiply the value by either 3, or the fuzzy logical complement of 3. Remember that our sequence space is closely mapped to a ternary number. The shift function, when applied to a ternary number, is equivalent to multiplication by a factor of 3. So,

$$S(3x) = \sigma(S(x))$$

We still have not demonstrated a conjugate function, as we have not considered the case where $\frac{2}{3} \leq x \leq 1$. In this case, we must take the logical complement of our value before performing the shift. Rather than incorporating this behaviour into σ, we instead incorporate it into our itinerary function, S. We define a new itinerary function, S', as:

$$S'(0.a_1a_2a_3\ldots) = \begin{cases} a_1^{'}a_2^{'}a_3^{'}\ldots & \text{if } a_1 = 0 \\ \bar{a_1}^{'}\bar{a_2}^{'}\bar{a_3}^{'}\ldots & \text{if } a_1 = 2 \end{cases}$$

where

$$a_i^{'} = \begin{cases} 0 & \text{if } a_i = 0 \\ 1 & \text{if } a_i = 2 \end{cases}$$

and

$$\bar{a_i}' = \begin{cases} 1 & \text{if } a_i = 0 \\ 0 & \text{if } a_i = 2 \end{cases}$$

For example: $S(0.2022202020202020\ldots) = 010001010101010\ldots$, while $S(0.0022002200220\ldots) = 0011001100110011\ldots$.

Homeomorphism

We now have enough information to make the following statement:

Theorem 5.9: The shift map σ on Σ' is conjugate to f on K'.

Proof. We know by Lemma 5.1 that $f : K' \to K'$. We have now to show that there exists an homeomorphism $S' : K' \to \Sigma'$ such that $S \circ f(x) = \sigma \circ S'(x)$ for any $x \in K'$; that is, the following diagram commute:

$$\begin{array}{ccccc} & & f & & \\ & K' & \longrightarrow & K' & \\ S' & \downarrow & & \downarrow & S' \\ & & \sigma & & \\ & \Sigma' & \longrightarrow & \Sigma' & \end{array}$$

Note: Real numbers in K' will be represented in their ternary expansion.

It is easy to see that S' is a well defined homeomorphism.

Consider $x \in K'$. By the properties of K', we can write x as a ternary expansion not containing any 1 and not terminating with infinite 0s nor infinite 2s. Let $x = 0.a_1a_2a_3\ldots$ with $a_i = 0$ or 2. Consider $f(x)$. By definition of f, and by definition

of ternary expansion, if $a_1 = 0$ then $x \in (0, \frac{1}{3})$ and thus $f(x) = 3x$ and its ternary expansion can be written as $0.a_2a_3\ldots$. On the other hand, if $a_1 = 2$ then $x \in (\frac{2}{3}, 1)$, thus $f(x) = 3(1-x)$ and its ternary expansion can be written as $0.\bar{a_2}\bar{a_3}\ldots$. in other words, we have:

$$f(x) = \begin{cases} 0.a_2a_3\ldots & \text{if } a_1 = 0 \\ 0.\bar{a_2}\bar{a_3}\ldots & \text{if } a_1 = 2 \end{cases} \tag{5.2}$$

By definition of S' we have that $S' \circ f(x)$ is the sequence of digits of $f(x)$ to the right of the radix, where every 2 is replaced by 1 when $a_1 = 0$, and every 2 is replaced by 0 and every 0 by 1 when $a_1 = 2$,.

Likewise, we know that $\sigma \circ S'(x)$ x applies $S'(x)$ to x, then shifts it to the left, which produces an identical binary sequence to the one above.

Therefore, $S' \circ f(x) \equiv \sigma \circ S'(x)$, and σ on Σ' is conjugate to f on K'.

□

Chaos

To prove that class CH4 rules are chaotic, it is sufficient to show that the shift map over the binary sequence space is chaotic. While this has been well established [16], we have subtly altered the homeomorphic space Σ, so we will now establish that this proof remains valid.

Before we begin, we define a distance metric d on Σ', given by:

$$d[s,t] = \sum_{i=0}^{\infty} \frac{|s_i - t_i|}{2^i}$$

This metric is the classical metric defined over binary sequence spaces. Note that

more significance is given to differences in digits toward the beginning of the sequence. So, differences in the most significant bit will have the greatest contribution to the distance metric, while differences in each digit thereafter will contribute exponentially less to the distance than the one that preceded it.

Theorem 5.10: The shift map σ on Σ' is chaotic.

Proof. Recall that Σ' is the set of all binary sequences that do not terminate with infinite 0s or infinite 1s, and σ is the shift map.

To show that σ on Σ' is chaotic, it is sufficient to show:

1. Sensitivity to initial conditions
2. Density of periodic orbits
3. Topological mixing

We demonstrate each of these properties below:

Sensitivity to initial conditions

Let $\beta = 1$. For any $s \in \Sigma'$ and $\epsilon > 0$, choose n so that $\frac{1}{2^n} < \epsilon$. Suppose $t \in \Sigma'$ satisfies $d[s,t] < \frac{1}{2^n}$ but $t \neq s$. Then we know that $t_i = s_i$ for $i = 0, \ldots, n$. However, since $t \neq s$, there exists $k > n$ such that $s_k \neq t_k$. So, $|s_k - t_k| = 1$. Now consider the sequences $\sigma^k(s)$ and $\sigma^k(t)$. The initial entries of each of these sequences are different, so we have

$$d[\sigma^k(s), \sigma^k(t)] \geq \frac{|s_k - t_k|}{2^0} + \sum_{i=0}^{\infty} \frac{0}{2^i} = 1$$

.

Therefore, the shift is sensitive to initial conditions.

Density of periodic orbits

Consider the point $\hat{s} = (0\,1\ 00\,01\,10\,11\ 000\,001\ \ldots)$. That is, $\hat{s}$ is the sequence which consists of all possible clocks of 0s and 1s of length 1, followed by all such blocks of length 2, then length 3, etc.

Now, choose any arbitrary point $s = (s_0 s_1 s_2 \ldots) \in \Sigma'$, choose $\epsilon > 0$, and choose n so that $\frac{1}{2^n} < \epsilon$. By definition, at some position k in $\hat{s}$ lies the digits $s_0 s_1 s_2 \ldots s_n$. If you apply the shift map σ k times to $\hat{s}$, then the first $n+1$ entries of $o^k(\hat{s})$ are $s_0 s_1 s_2 \ldots s_n$. By the Proximity Theorem,

$$d[\sigma^k(\hat{s}), s] \leq \frac{1}{2^n} < \epsilon$$

Therefore, the point $\hat{s}$ has an orbit that forms a dense subset of Σ'.

Topological mixing

A dynamical system is *transitive* if for any pair of points x and y, and any $\epsilon > 0$, there is a third point z within ϵ of x whose orbit comes within ϵ of y. Clearly, since Σ' has an orbit that forms a dense subset of Σ', then Σ' is topologically mixing, since $\hat{s}$ comes arbitrarily close to all points.

All three criteria for chaos have been met, therefore the shift map σ on Σ' is chaotic. □

Now we can finally conclude that:

Theorem 5.11: Elementary fuzzy cellular automata, in conjunctive normal form, with even rule numbers less than 128, and with homogeneous initial conditions, defined by the function $f(x,x,x) = min\{1,(3-3x)\} \cdot min\{1,(3x)\}$, is chaotic over the open

Cantor set.

Proof. We have already shown that $S' \circ f(x) \equiv \sigma \circ S'(x)$, and σ on Σ' is conjugate to f on K'. We have also shown that the shift map σ on Σ' is chaotic. Therefore class CH4 rules are chaotic over K'.

□

5.3 Classification with Random Initial Configuration

With a random initial configuration, we were unable to find any canonical classification, so we provide an empirical classification instead. For each class we provide a name for the class, give the pattern of its binary representation, provide lists of necessary and forbidden factors required to produce the observed behaviour. This is followed by the rule that produces this behaviour with the fewest number of factors (simplest case). This is then followed by a description of the observed behaviour, image(s) of the observed behaviour, and an analysis of what we believe causes the observed behaviour.

Simplifying the local transition function is difficult due to the nature of the conjunctive normal form, and the presence of the *min* function with each factor. To better understand the observed behaviour, the analysis will usually include a table showing the formula for the local transition function under certain conditions (e.g. a homogeneous local configuration). In many cases, we can eliminate factors of the form $min\{1, x\}$, where $x \geq 1$, since this factor will equal 1. For example, the factor $min\{1, (-x+2)\}$ can be simplified to 1, since any input variable of a FCA must have a value in the range $[0, 1]$, thus $-x + 2 \geq 1$.

The binary representation of a class of rules uses the same binary representation as an individual rule, with the addition of a third digit, $*$, where $* \in \{0, 1\}$, known as a "don't care" state.

For our colour mapping, we use the spectrum colour map with white and black boundaries, as discussed in section 3.2. Note that to increase clarity for printing, we have chosen to have zero be black, and one be white. For each simulation, we also create a bitmap image showing which cells that have the value of exactly 1.0, called the "ones" image. Cells with a value of 1.0 appear black, and cells that are < 1.0 appear white. This can help us understand the behaviour of certain CA where the $min(1, x)$ function becomes a major factor in its evolution.

5.3.1 CFMS Random Class 1 (CR1)

Name	Class CR1
Binary	0******0
Necessary Factors	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$
	$min\{1, (x_1 + x_2 + x_3)\}$
Forbidden Factors	None
Simplest Case	r_{126}
Rules	$r_x \vert x < 128, x \bmod 2 = 0$

Class CR1 is equivalent to class CH4 in the homogeneous case. Class CR1 exhibits interesting behaviour. Class CR1 rules do not exhibit any obvious periodicity. They appear to produce pseudo-random, or even possibly chaotic, behaviour.

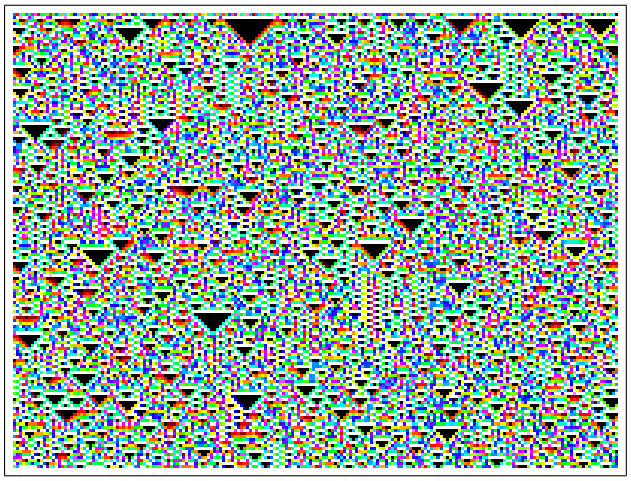


Figure 5.6: Rule 126

5.3.2 CFMS Random Class 2 (CR2)

Name	Class CR2
Binary	0******1
Necessary Factors	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$
Forbidden Factors	$min\{1, (x_1 + x_2 + x_3)\}$
Simplest Case	r_{127}
Rules	$r_x \vert x < 128, x \bmod 2 = 1$

Figure 5.7: Rule 127

Class CR2 is equivalent to class CH3 in the homogeneous case. Class CR2 rules exhibit several different behaviours.

5.3.3 CFMS Random Class 3 (CR3)

Name	Class CR3
Binary	10**0***
Necessary Factors	$min\{1, (-x_1 - x_2 + x_3 + 2)\}$
	$min\{1, (x_1 - x_2 - x_3 + 2)$
Forbidden Factors	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$
Simplest Case	r_{183}
Rules	r_{128}, r_{129}, r_{130}, r_{131}, r_{132}, r_{133}, r_{134}, r_{135}, r_{144}, r_{145}, r_{146}, r_{147}, r_{148}, r_{149}, r_{150}, r_{151}, r_{160}, r_{161}, r_{162}, r_{163}, r_{164}, r_{165}, r_{166}, r_{167}, r_{176}, r_{177}, r_{178}, r_{179}, r_{180}, r_{181}, r_{182}, r_{183}

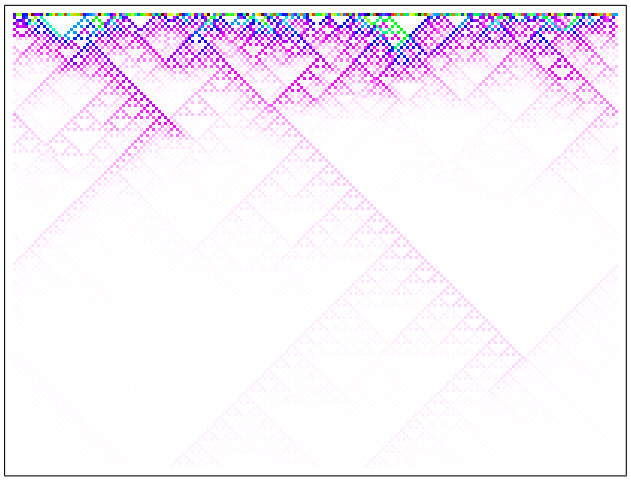


Figure 5.8: Rule 183

Class CR3 is a subset of class CH1 $\cup$ CH2 in the homogeneous case. Class CR3 exhibits striking behaviour. Rules in Class CR3 with a random initial configuration produce Sierpinski triangles, with the white cells representing a value of 1 and the

magenta cells representing values approaching 1.

Class CR3 exhibits periodicity both spatially and temporally. Spatially, they exhibit the configuration $X^t = (1, a)^{n/2}$, where $a \in [0, 1]$, i.e. cells alternate between 1 and other values. Temporally, the configuration alternates between $X^t = (1, a)^{n/2}$ and $X^t = (a, 1)^{n/2}$, i.e. cells that were 1 in the previous generation become less than 1, and *vice versa*. This can be observed clearly in the bitmap "ones" diagram shown in figure 5.9, where values of 1 are shown in white. This produces a checker-board pattern. The black lines appear as interference between adjacent blocks of periodicity, where two patterns are established out of phase (e.g., one part of the row will have 1s in even cells, while another has 1s in odd cells). The value of adjacent cells determines whether this line moves left or right with time. We conjecture that with an an even size CA, these lines will always resolve themselves, while an odd sized CA will always be left with one such line.

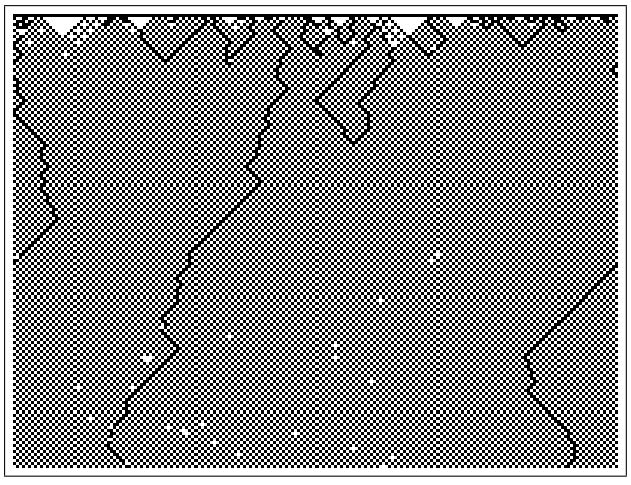


Figure 5.9: Rule 183 values of 1

We can understand some of Class CR3 behaviour by examining the simplest case of Rule 183. The local transition function is shown below in table 5.10 for several input cases.

$(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$	$\boldsymbol{f_{183}}(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$
(x, y, z)	$min\{1, (-x - y + z + 2)\} \cdot min\{1, (x - y - z + 2)\}$
(x, x, x)	1
$(1, 1, 1)$	1
$(1, x, 1)$	1
$(x, 1, 1)$	x
$(1, 1, x)$	x
$(x, 1, y)$	$min\{1, (-x + y + 1)\} \cdot min\{1, (x - y + 1)\}$
$(x, 1, y) \vert x < y$	$x - y + 1$
$(x, 1, y) \vert x > x$	$-x + y + 1$
$(x, 1, y) \vert x = y$	1

Table 5.10: Several input cases of CFMS CNF Rule 183

Observe that $(1, x, 1)$ goes to 1, while (x, 1, y) goes to ≤ 1. This is responsible for maintaining the checker-board pattern we see in the ones diagram. Observe also that as x tends to 1, $(x, 1, y)$ tends to y, and as y tends to 1, $(x, 1, y)$ tends to x. This is similar to the inverse of the behaviour of boolean rule 90, where $(0, 0, 1) \rightarrow 1$, and $(1, 0, 0) \rightarrow 1$. In this case, the value of < 1 is equivalent to 1 in the behaviour of boolean rule 90.

5.3.4 CFMS Random Class 4a (CR4a)

Name	Class CR4a
Binary	$10{*}{*}1{*}{*}{*}$
Necessary Factors	$min\{1, (-x_1 - x_2 + x_3 + 2)\}$
Forbidden Factors	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$
	$min\{1, (x_1 - x_2 - x_3 + 2)\}$
Simplest Case	r_{191}
Rules	r_{136}, r_{137}, r_{138}, r_{139}, r_{140}, r_{141}, r_{142}, r_{143}, r_{152}, r_{153}, r_{154}, r_{155}, r_{156}, r_{157}, r_{158}, r_{159}, r_{168}, r_{169}, r_{170}, r_{171}, r_{172}, r_{173}, r_{174}, r_{175}, r_{184}, r_{185}, r_{186}, r_{187}, r_{188}, r_{189}, r_{190}, r_{191}

Class CR4a is a subset of class CH1 $\cup$ CH2 in the homogeneous case. Class CR4a exhibits periodicity temporally, in the form of a shift.

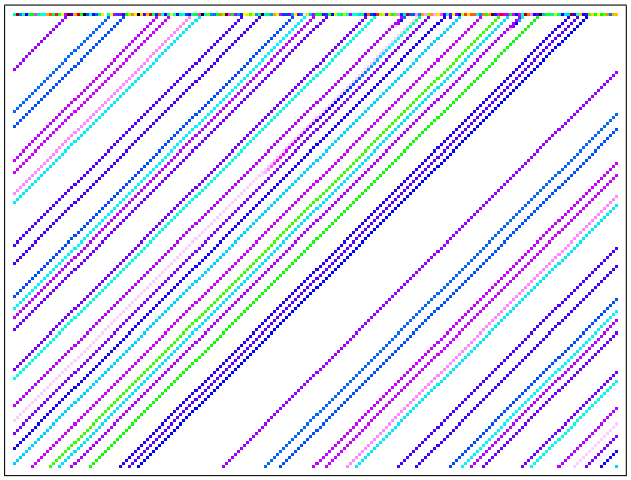


Figure 5.10: Rule 191

We can understand some of Class CR4a behaviour by examining the simplest case of Rule 191. The local transition function is shown below in table 5.11 for several input cases.

In the first generation, most cells go to 1, which some remain < 1. The shifting

$(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$	$\boldsymbol{f_{191}}(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$
(x, y, z)	$min\{1, (-x - y + z + 2)\}$
(x, x, x)	1
$(1, 1, 1)$	1
$(1, x, 1)$	1
$(x, 1, 1)$	1
$(1, 1, x)$	x

Table 5.11: Several input cases of CFMS CNF Rule 191

behaviour becomes apparent when observing that $(1, 1, x) \rightarrow x$, while most other local configurations go to 1.

5.3.5 CFMS Random Class 4b (CR4b)

Name	Class CR4b
Binary	$11{*}{*}0{*}{*}{*}$
Necessary Factors	$min\{1, (x_1 - x_2 - x_3 + 2)\}$
Forbidden Factors	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$
	$min\{1, (-x_1 - x_2 + x_3 + 2)\}$
Simplest Case	r_{247}
Rules	r_{192}, r_{193}, r_{194}, r_{195}, r_{196}, r_{197}, r_{198}, r_{199}, r_{208}, r_{209}, r_{210}, r_{211}, r_{212}, r_{213}, r_{214}, r_{215}, r_{224}, r_{225}, r_{226}, r_{227}, r_{228}, r_{229}, r_{230}, r_{231}, r_{240}, r_{241}, r_{242}, r_{243}, r_{244}, r_{245}, r_{246}, r_{247}

Class CR4b is a subset of class CH1 $\cup$ CH2 in the homogeneous case. Class CR4b exhibits periodicity temporally, in the form of a shift.rally, in the form of a shift.

We can understand some of Class CR4b behaviour by examining the simplest case of Rule 247. The local transition function is shown below in table 5.12 for several

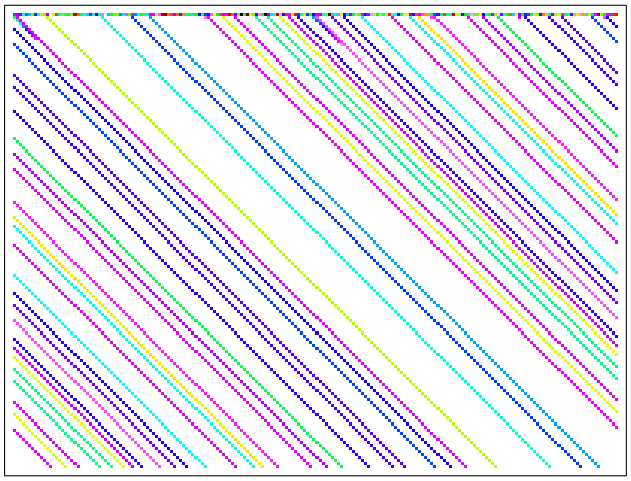


Figure 5.11: Rule 247

input cases:

$(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$	$\boldsymbol{f_{247}}(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$
(x, y, z)	$min\{1, (x - y - z + 2)\}$
(x, x, x)	1
$(1, 1, 1)$	1
$(1, x, 1)$	1
$(x, 1, 1)$	x
$(1, 1, x)$	1

Table 5.12: Several input cases of CFMS CNF Rule 247

In the first generation, most cells go to 1, which some remain < 1. The shifting behaviour becomes apparent when observing that $(x, 1, 1) \rightarrow x$, while most other local configurations go to 1. Class CR4b is a reflection of Class CR4a.

5.3.6 CFMS Random Class 5 (CR5)

Name	Class CR5
Binary	$110*1***$
Necessary Factors	$min\{1,(-x_1+x_2-x_3+2)\}$
Forbidden Factors	$min\{1,(-x_1-x_2-x_3+3)\}$
	$min\{1,(-x_1-x_2+x_3+2)\}$
	$min\{1,(x_1-x_2-x_3+2)\}$
Simplest Case	r_{223}
Rules	r_{200}, r_{201}, r_{202}, r_{203}, r_{204}, r_{205}, r_{206}, r_{207}, r_{216}, r_{217}, r_{218},
	r_{219}, r_{220}, r_{221}, r_{222}, r_{223}

Class CR5 is a subset of class CH1 $\cup$ CH2 in the homogeneous case. Class CR5 exhibits periodicity temporally, in the form of a fixed point configuration.

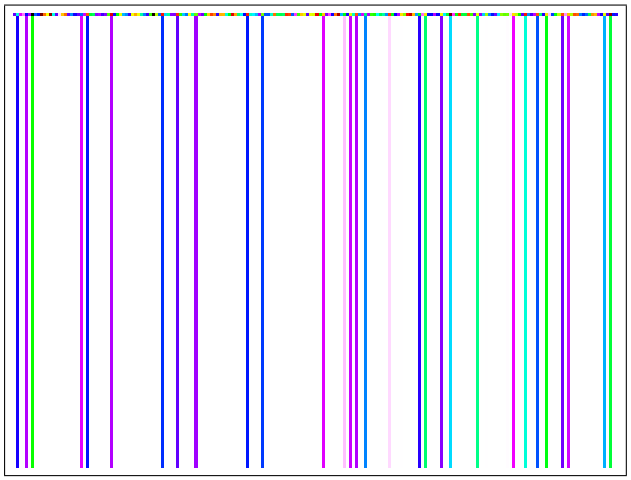


Figure 5.12: Rule 223

We can understand some of Class CR5's fixed point behaviour by examining the simplest case of Rule 223. The local transition function is shown below in table 5.13 for several input cases:

In the first generation, most cells go to 1, which some remain < 1. The fixed

$(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$	$\boldsymbol{f_{223}}(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3})$
(x, y, z)	$min\{1, (-x + y - z + 2)\}$
(x, x, x)	1
$(1, 1, 1)$	1
$(1, x, 1)$	x
$(x, 1, 1)$	1
$(1, 1, x)$	1

Table 5.13: Several input cases of CFMS CNF Rule 223

point behaviour becomes apparent when observing that $(1, x, 1) \rightarrow x$, while most other local configurations go to 1.

5.3.7 CFMS Random Class 6 (CR6)

Name	Class CR6
Binary	$111*1***$
Necessary Factors	1
Forbidden Factors	$min\{1, (-x_1 - x_2 - x_3 + 3)\}$
	$min\{1, (-x_1 - x_2 + x_3 + 2)\}$
	$min\{1, (-x_1 + x_2 - x_3 + 2)\}$
	$min\{1, (x_1 - x_2 - x_3 + 2)\}$
Simplest Case	r_{255}
Rules	r_{232}, r_{233}, r_{234}, r_{235}, r_{236}, r_{237}, r_{238}, r_{239}, r_{248}, r_{249}, r_{250},
	r_{251}, r_{252}, r_{253}, r_{254}, r_{255}

Class CR6 is a subset of class CH1 $\cup$ CH2 in the homogeneous case. Class CR6 exhibits periodicity temporally, in the form of a fixed point configuration. Starting from a random initial configuration, all values converge to 1.

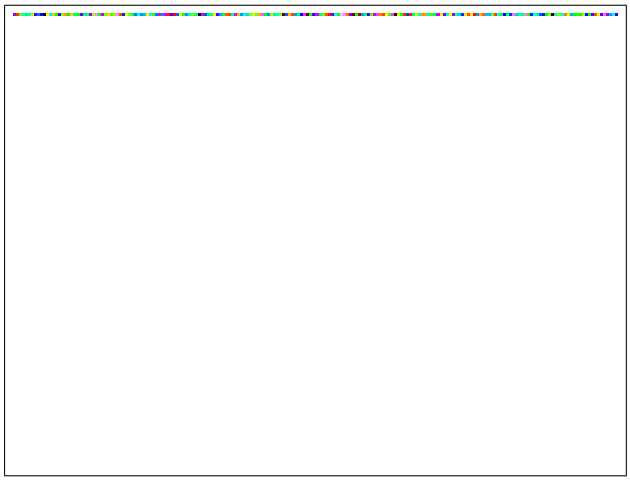


Figure 5.13: Rule 255

5.4 Conclusion

We began our investigation with the realization that FCA CNF "fuzzification", being a conjunction of disjunctions, produce formulae with too many terms to be easily analyzed. In order to facilitate analysis, we began by restricting our investigation to the special case of a homogeneous initial configuration.

This allowed us to analyze FCA in CNF in this particular case, in which we discovered that all 256 such FCA fall into one of four classes. We were able to solve the asymptotic behaviour of each of these classes.

The first three classes exhibited simplistic dynamics: The first class, CH1, converges directly to one. The second class, CH2, has two fixed points, and converges either to zero or one. The third class, CH3, has a fixed point at $\frac{3}{4}$, but all other values will converge to a temporally periodic configuration that alternates between zero and one.

The final class, CH4, to our surprise, exhibited chaos. We demonstrated first that CH4 rules have both fixed point and periodic configurations. We then observed

that certain points exhibit a periodic behaviour with a period of three, and that points that converge to this cycle can be defined recursively. We then show that this recursive definition produces points that will always lie on the *open Cantor set*. Finally, we were then able to prove the existence of chaos using symbolic dynamics. The presence of chaos was surprising, since it has been proven that no such chaos exists in FCA in DNF.

In the second half of this chapter, we simulated all 256 rules in the general case, with a random initial configuration, so that we could classify them empirically. We observed that all rules fall into one of six classes of behaviour. The first two classes, CR1, and CR2, exactly match the classes CH3 and CH4, respectively. All other classes do not appear to have a correlation with the classes seen in the homogeneous case.

For each class, we provided the pattern of its binary representation, lists of necessary and forbidden factors required to produce the observed behaviour, and the rule that produces the behaviour with the fewest number of factors. We then include an image that is representative of the asymptotic behaviour seen in that class, and conjecture as to the reason for the observed behaviour.

In this chapter, we have chosen to study FCA in CNF with CFMS logic. In the next chapter, we shall study how the choice of fuzzy logic affects the results of our analysis.

Chapter 6

Fuzzy CA in Conjunctive Normal Form with Other Logics

In this chapter, we define fuzzy cellular automata in conjunctive normal form in five additional fuzzy logics: Probabilistic, Łukasiewicz, Gödel, Zadeh, and Product. In each case, we investigate the asymptotic behaviour of these CA analytically in the homogeneous case by solving them algebraically. We find that elementary FCA in CNF in these other logics have extremely simple behaviours that do not capture many of the properties of their Boolean counterparts.

Note in our analysis of the homogeneous case, our results are applicable to CA of both finite (circular) and infinite size.

6.1 Probabilistic

6.1.1 Definitions

Probabilistic logic is similar to CFMS logic, except the definition of disjunction has changed. Recall the definition of Probabilistic fuzzification:

Boolean	Fuzzy
$\neg x$	$1 - x$
$x \wedge y$	$x \cdot y$
$x \vee y$	$x + y - x \cdot y$

Table 6.1: Probabilistic Fuzzification

As an exercise, consider rule $183 = 128 + 32 + 16 + 4 + 2 + 1$:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (1, 0, 1, 1, 0, 1, 1, 1)$$

The conjunctive normal form expression of rule 183 is:

$$f_{183}(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$$

The fuzzification process yields:

$$(x_1x_2x_3 - x_1x_2 + 1) \cdot (x_1x_2x_3 - x_2x_3 + 1)$$

To simplify the fuzzification process, table 6.2 may be used in the construction of the local transition function of an elementary CNF FCA using Probabilistic logic.

Since the conjunction operator is represented as multiplication in Probabilistic logic, the presence of a 0 in a particular column of the rule's binary representation

Column	Value	Transition	Factor
0	1	$(0,0,0) \to 0$	$x_1x_2x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1 + x_2 + x_3$
1	2	$(0,0,1) \to 0$	$x_1x_3 + x_2x_3 - x_1x_2x_3 - x_3 + 1$
2	4	$(0,1,0) \to 0$	$x_1x_2 + x_2x_3 - x_1x_2x_3 - x_2 + 1$
3	8	$(0,1,1) \to 0$	$x_1x_2x_3 - x_2x_3 + 1$
4	16	$(1,0,0) \to 0$	$x_1x_2 + x_1x_3 - x_1x_2x_3 - x_1 + 1$
5	32	$(1,0,1) \to 0$	$x_1x_2x_3 - x_1x_3 + 1$
6	64	$(1,1,0) \to 0$	$x_1x_2x_3 - x_1x_2 + 1$
7	128	$(1,1,1) \to 0$	$1 - x_1x_2x_3$

Table 6.2: FCA Probabilistic CNF Factors

indicates the presence of the given factor in its transition function. (Columns are counted from the right, starting with column 0.) Rule 183 has the binary number $(1,0,1,1,0,1,1,1)$, which has two zeros: in column 3 and column 6. So, the Probabilistic fuzzification of Rule 183 in CNF is simply the product of the two factors indicated in table 6.2 for those columns.

6.1.2 Homogeneous Initial Configuration

Recall from table 6.2 that the local transition function $f(x_1, x_2, x_3)$ of a FCA in CNF is a product of factors whose presence (or absence) is determined by the presence (or absence) of transitions that go to 0. In the case of a homogeneous configuration, $x_1 = x_2 = x_3 = x$, so we can simplify the products from table 6.2. The result is shown below in table 6.3.

6.2 Łukasiewicz

Łukasiewicz logic is similar to CFMS logic, except the definition of conjunction has changed. Recall the definition of Łukasiewicz fuzzification:

Transition	**CNF Factors**	**Factors When** $\boldsymbol{x_1 = x_2 = x_3 = x}$
$(0,0,0) \to 0$	$x_1x_2x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1 + x_2 + x_3$	$x^3 - 3x^2 + 3x$
$(0,0,1) \to 0$	$x_1x_3 + x_2x_3 - x_1x_2x_3 - x_3 + 1$	$-x^3 + 2x^2 - x + 1$
$(0,1,0) \to 0$	$x_1x_2 + x_2x_3 - x_1x_2x_3 - x_2 + 1$	$-x^3 + 2x^2 - x + 1$
$(0,1,1) \to 0$	$x_1x_2x_3 - x_2x_3 + 1$	$x^3 - x^2 + 1$
$(1,0,0) \to 0$	$x_1x_2 + x_1x_3 - x_1x_2x_3 - x_1 + 1$	$-x^3 + 2x^2 - x + 1$
$(1,0,1) \to 0$	$x_1x_2x_3 - x_1x_3 + 1$	$x^3 - x^2 + 1$
$(1,1,0) \to 0$	$x_1x_2x_3 - x_1x_2 + 1$	$x^3 - x^2 + 1$
$(1,1,1) \to 0$	$1 - x_1x_2x_3$	$1 - x^3$

Table 6.3: Homogeneous Probabilistic FCA CNF Factors

Boolean	Fuzzy
$\neg x$	$1 - x$
$x \wedge y$	$max\{0, x + y - 1\}$
$x \vee y$	$min\{1, x + y\}$

Table 6.4: Łukasiewicz Fuzzification

As an exercise, consider rule 183 = 128 + 32 + 16 + 4 + 2 + 1:

$$(111, 110, 101, 100, 011, 010, 001, 000) \to (1, 0, 1, 1, 0, 1, 1, 1)$$

The conjunctive normal form expression of rule 183 is:

$$f_{183}(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$$

The fuzzification process yields:

$$max\{0, min\{1, (-x_1 - x_2 + x_3 + 2)\} + min\{1, (x_1 - x_2 - x_3 + 2)\} - 1\}$$

Since Łukasiewicz uses the same definition for disjunction as CFMS, table 5.4 may be used in the construction of the local transition function of an elementary

CNF FCA using CFMS logic. However, unlike CFMS, you cannot simply multiply these terms together to perform conjunction. The formula $max\{0, x+y-1\}$ must be used instead for conjunction, making the construction of the local transition function difficult.

6.3 Gödel

6.3.1 Definitions

Recall the definition of Gödel fuzzification:

Boolean	Fuzzy
$\neg x$	if $x = 0$ then 1, else 0
$x \wedge y$	$min\{x, y\}$
$x \vee y$	$max\{x, y\}$

Table 6.5: Gödel Fuzzification

Note the definition for negation in Gödel logic: *if* $x = 0$ *then 1, else 0.* An equivalent way of writing this is $1 - \lceil x \rceil$, where $\lceil\ \rceil$ denotes the ceiling function. Unlike the definition of negation we have seen before, this definition of negation is discontinuous. This means that Gödel fuzzy logic functions as a whole cannot be considered continuous.

As an exercise, consider rule $183 = 128 + 32 + 16 + 4 + 2 + 1$:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (1, 0, 1, 1, 0, 1, 1, 1)$$

The conjunctive normal form expression of rule 183 is:

$$f_{183}(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$$

The fuzzification process yields:

$$min[max(1 - \lceil x_1 \rceil, 1 - \lceil x_2 \rceil, x_3),\ max(x_1, 1 - \lceil x_2 \rceil, 1 - \lceil x_3 \rceil)]$$

To simplify the fuzzification process, table 6.6 may be used in the construction of the local transition function of an elementary CNF FCA using Gödel logic.

Column	Value	Transition	Term
0	1	$(0,0,0) \to 0$	$max\{x_1, x_2, x_3\}$
1	2	$(0,0,1) \to 0$	$max\{x_1, x_2, 1 - \lceil x_3 \rceil\}$
2	4	$(0,1,0) \to 0$	$max\{x_1, 1 - \lceil x_2 \rceil, x_3\}$
3	8	$(0,1,1) \to 0$	$max\{x_1, 1 - \lceil x_2 \rceil, 1 - \lceil x_3 \rceil\}$
4	16	$(1,0,0) \to 0$	$max\{1 - \lceil x_1 \rceil, x_2, x_3\}$
5	32	$(1,0,1) \to 0$	$max\{1 - \lceil x_1 \rceil, x_2, 1 - \lceil x_3 \rceil\}$
6	64	$(1,1,0) \to 0$	$max\{1 - \lceil x_1 \rceil, 1 - \lceil x_2 \rceil, x_3\}$
7	128	$(1,1,1) \to 0$	$max\{1 - \lceil x_1 \rceil, 1 - \lceil x_2 \rceil, 1 - \lceil x_3 \rceil\}$

Table 6.6: FCA Gödel CNF Terms

The presence of a 0 in a particular column of the rule's binary representation indicates the presence of the given term in its transition function. (Columns are counted from the right, starting with column 0.) The final formula for the rule is simply the minimum of all of the present terms. Rule 183 has the binary number $(1,0,1,1,0,1,1,1)$, which has two zeros: in column 3 and column 6. So, the Gödel fuzzification of Rule 183 in CNF is the minimum of the two terms indicated in table 6.6 for those columns.

6.3.2 Homogeneous Initial Configuration

When dealing with a homogeneous initial configuration, we can exploit the fact that $min\{x, x, \ldots x\} = max\{x, x, \ldots x\} = x$, as well as the fact that the negation function always produces an extreme value of either 0 or 1. The result is shown below in table 6.7.

Transition	**CNF Terms**	**Terms When $x_1 = x_2 = x_3 = x$**	
		$x > 0$	$x = 0$
$(0,0,0) \to 0$	$max\{x_1, x_2, x_3\}$	x	x
$(0,0,1) \to 0$	$max\{x_1, x_2, 1 - \lceil x_3 \rceil\}$	x	1
$(0,1,0) \to 0$	$max\{x_1, 1 - \lceil x_2 \rceil, x_3\}$	x	1
$(0,1,1) \to 0$	$max\{x_1, 1 - \lceil x_2 \rceil, 1 - \lceil x_3 \rceil\}$	x	1
$(1,0,0) \to 0$	$max\{1 - \lceil x_1 \rceil, x_2, x_3\}$	x	1
$(1,0,1) \to 0$	$max\{1 - \lceil x_1 \rceil, x_2, 1 - \lceil x_3 \rceil\}$	x	1
$(1,1,0) \to 0$	$max\{1 - \lceil x_1 \rceil, 1 - \lceil x_2 \rceil, x_3\}$	x	1
$(1,1,1) \to 0$	$max\{1 - \lceil x_1 \rceil, 1 - \lceil x_2 \rceil, 1 - \lceil x_3 \rceil\}$	0	1

Table 6.7: Homogeneous Gödel FCA CNF Terms

Remember that the final formula is the minimum of the terms shown above where the transition function goes to 0. As we can see, in the absence of the transition $(1,1,1) \to 0$ a homogeneous configuration of $X = x^n$, where $x > 0$ is a fixed point configuration.

6.4 Zadeh

6.4.1 Definitions

Recall the definition of Zadeh fuzzification:

Boolean	Fuzzy
$\neg x$	$1 - x$
$x \wedge y$	$min\{x, y\}$
$x \vee y$	$max\{x, y\}$

Table 6.8: Zadeh Fuzzification

Zadeh logic has the same behaviour as Gödel logic, except the standard definition for negation that we have seen before.

As an exercise, consider rule $183 = 128 + 32 + 16 + 4 + 2 + 1$:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (1, 0, 1, 1, 0, 1, 1, 1)$$

The conjunctive normal form expression of rule 183 is:

$$f_{183}(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$$

The fuzzification process yields:

$$min[max(1 - x_1, 1 - x_2, x_3),\ max(x_1, 1 - x_2, 1 - x_3)]$$

To simplify the fuzzification process, table 6.9 may be used in the construction of the local transition function of an elementary CNF FCA using Zadeh logic.

The presence of a 0 in a particular column of the rule's binary representation indicates the presence of the given term in its transition function. (Columns are counted from the right, starting with column 0.) The final formula for the rule is simply the minimum of all of the present terms. Rule 183 has the binary number $(1, 0, 1, 1, 0, 1, 1, 1)$, which has two zeros: in column 3 and column 6. So, the Gödel fuzzification of Rule 183 in CNF is the minimum of the two terms indicated in table

Column	Value	Transition	Term
0	1	$(0,0,0) \to 0$	$max\{x_1, x_2, x_3\}$
1	2	$(0,0,1) \to 0$	$max\{x_1, x_2, 1-x_3\}$
2	4	$(0,1,0) \to 0$	$max\{x_1, 1-x_2, x_3\}$
3	8	$(0,1,1) \to 0$	$max\{x_1, 1-x_2, 1-x_3\}$
4	16	$(1,0,0) \to 0$	$max\{1-x_1, x_2, x_3\}$
5	32	$(1,0,1) \to 0$	$max\{1-x_1, x_2, 1-x_3\}$
6	64	$(1,1,0) \to 0$	$max\{1-x_1, 1-x_2, x_3\}$
7	128	$(1,1,1) \to 0$	$max\{1-x_1, 1-x_2, 1-x_3\}$

Table 6.9: FCA Zadeh CNF Terms

6.9 for those columns.

6.4.2 Homogeneous Initial Configuration

When dealing with a homogeneous initial configuration, we can exploit the fact that $min\{x, x, \ldots x\} = max\{x, x, \ldots x\} = x$. The result is shown below in table 6.10.

Transition	**CNF Terms**	**Terms When $x_1 = x_2 = x_3 = x$**	
		$x < 0.5$	$x \geq 0.5$
$(0,0,0) \to 0$	$max\{x_1, x_2, x_3\}$	x	x
$(0,0,1) \to 0$	$max\{x_1, x_2, 1-x_3\}$	1-x	x
$(0,1,0) \to 0$	$max\{x_1, 1-x_2, x_3\}$	1-x	x
$(0,1,1) \to 0$	$max\{x_1, 1-x_2, 1-x_3\}$	1-x	x
$(1,0,0) \to 0$	$max\{1-x_1, x_2, x_3\}$	1-x	x
$(1,0,1) \to 0$	$max\{1-x_1, x_2, 1-x_3\}$	1-x	x
$(1,1,0) \to 0$	$max\{1-x_1, 1-x_2, x_3\}$	1-x	x
$(1,1,1) \to 0$	$max\{1-x_1, 1-x_2, 1-x_3\}$	1-x	1-x

Table 6.10: Homogeneous Zadeh FCA CNF Terms

Remember that the final formula is the minimum of the terms shown above where the transition function goes to 0. As we can see, in the absence of the transition $(0,0,0) \to 0$ and $(1,1,1) \to 0$, a homogeneous configuration will be temporally

periodic, with period 2:

If $X^t = (x)^n$, then $\forall m \geq 0, X^{t+2m} = (x)^n$ and $X^{t+2m+1} = (1-x)^n$.

6.5 Product

6.5.1 Definitions

Product logic is similar to Probabilistic logic, except the definition of negation is the same as Gödel logic. Recall the definition of Product fuzzification:

Boolean	Fuzzy
$\neg x$	if $x = 0$ then 1, else 0
$x \wedge y$	$x \cdot y$
$x \vee y$	$x + y - x \cdot y$

Table 6.11: Product Fuzzification

As an exercise, consider rule $183 = 128 + 32 + 16 + 4 + 2 + 1$:

$$(111, 110, 101, 100, 011, 010, 001, 000) \rightarrow (1, 0, 1, 1, 0, 1, 1, 1)$$

The conjunctive normal form expression of rule 183 is:

$$f_{183}(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$$

The fuzzification process yields:

$$(\lceil x_1 \rceil \lceil x_2 \rceil (x_3 - 1) + 1) \cdot (\lceil x_2 \rceil \lceil x_3 \rceil (x_1 - 1) + 1)$$

To simplify the fuzzification process, table 6.12 may be used in the construction

of the local transition function of an elementary CNF FCA using Product logic.

Column	Value	Transition	Factor
0	1	$(0,0,0) \to 0$	$x_1x_2x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1 + x_2 + x_3$
1	2	$(0,0,1) \to 0$	$\lceil x_3 \rceil (x_1 + x_2 - x_1x_2 - 1) + 1$
2	4	$(0,1,0) \to 0$	$\lceil x_2 \rceil (x_1 + x_3 - x_1x_3 - 1) + 1$
3	8	$(0,1,1) \to 0$	$\lceil x_2 \rceil \lceil x_3 \rceil (x_1 - 1) + 1$
4	16	$(1,0,0) \to 0$	$\lceil x_1 \rceil (x_2 + x_3 - x_2x_3 - 1) + 1$
5	32	$(1,0,1) \to 0$	$\lceil x_1 \rceil \lceil x_3 \rceil (x_2 - 1) + 1$
6	64	$(1,1,0) \to 0$	$\lceil x_1 \rceil \lceil x_2 \rceil (x_3 - 1) + 1$
7	128	$(1,1,1) \to 0$	$1 - \lceil x_1 \rceil \lceil x_2 \rceil \lceil x_3 \rceil$

Table 6.12: FCA Product CNF Terms

Since the conjunction operator is represented as multiplication in Product logic, the presence of a 0 in a particular column of the rule's binary representation indicates the presence of the given factor in its transition function. (Columns are counted from the right, starting with column 0.) Rule 183 has the binary number $(1,0,1,1,0,1,1,1)$, which has two zeros: in column 3 and column 6. So, the Product fuzzification of Rule 183 in CNF is simply the product of the two factors indicated in table 6.12 for those columns.

6.5.2 Homogeneous Initial Configuration

Recall from table 6.12 that the local transition function $f(x_1, x_2, x_3)$ of a FCA in CNF is a product of factors whose presence (or absence) is determined by the presence (or absence) of transitions that go to 0. In the case of a homogeneous configuration, $x_1 = x_2 = x_3 = x$, so we can simplify the products from table 6.12. The result is shown below in table 6.13.

Transition	**Products When** $\boldsymbol{x_1 = x_2 = x_3 = x}$		
	General	$\boldsymbol{x > 0}$	$\boldsymbol{x = 0}$
$(0,0,0) \to 0$	$x^3 - 3x^2 + 3x$		
$(0,0,1) \to 0$	$\lceil x \rceil (2x - x^2 - 1) + 1$	$2x - x^2$	1
$(0,1,0) \to 0$	$\lceil x \rceil (2x - x^2 - 1) + 1$	$2x - x^2$	1
$(0,1,1) \to 0$	$\lceil x \rceil (x - 1) + 1$	x	1
$(1,0,0) \to 0$	$\lceil x \rceil (2x - x^2 - 1) + 1$	$2x - x^2$	1
$(1,0,1) \to 0$	$\lceil x \rceil (x - 1) + 1$	x	1
$(1,1,0) \to 0$	$\lceil x \rceil (x - 1) + 1$	x	1
$(1,1,1) \to 0$	$1 - \lceil x \rceil$	0	1

Table 6.13: Homogeneous Product FCA CNF Factors

6.6 Conclusion

In this chapter, we have defined FCA in CNF with five additional fuzzy logics. In each case, we have defined the local transition function. We then simplified each local transition function in the homogeneous case. Some cases produce extremely simplistic transition functions that may be of little interest, while others may warrant further study.

The algebraic complexity of the transition functions of these logics in the heterogeneous case has prevented us from analyzing them. While we did perform some preliminary simulations in the heterogeneous case, we found that most cases converged to a fixed point, so we did not pursue these simulations further. This simplistic behaviour, combined with our inability to properly analyze their transition functions, has led us to support the continued use of CFMS logic in FCA. However, as we were not able to make a definitive analysis, we leave the question of whether these logics are of further use in FCA research as an open problem.

Chapter 7

Conclusion

Fuzzy cellular automata are of interest for their complex dynamics, and potential applications in other fields. Despite their straightforward structure composed of a lattice of cells with local interactions, their properties have shown to be quite complex and interesting. Studying their dynamics have given insight into their Boolean counterparts, and their properties have made them applicable to other fields, most notably modelling and simulation.

Our research was motivated by the fact that to date, all fuzzy cellular automata research has been conducted in the disjunctive normal form. We began our research with the goal of exploring whether studying FCA in conjunctive normal form was of any interest.

In order to study FCA in CNF, we first created a software tool specifically to simulate and visualize fuzzy CA. The software tool is very robust, and could have a broad range of applications that go far beyond the scope of this thesis. It is currently being used, for example, to study network decontamination by mobile agents. We hope that the tool, which we plan to release publicly, will be of some benefit to the

cellular automata research community.

Our analysis of FCA in CNF began with the realization that simplifying the local transition function in CNF would prove infeasible. We instead performed an analysis of the special case of a homogeneous configuration. We were able to prove several key properties of FCA in CNF in the homogeneous case, the most important of which is that FCA in CNF exhibit chaos. This is in striking contrast to FCA in DNF, which have been shown to not exhibit chaos.

We then continued our study by performing an empirical classification of FCA in CNF in the heterogeneous case. We succeeded in showing that all FCA in CNF in this case exhibit one of 6 possible behaviours (or a reflection of one of those behaviours). It is interesting to note, however, that these classes do not seem to correspond to the classes that exist in DNF.

What is not known, is whether the theorems we prove in the homogeneous case are applicable in the general case. In particular, does chaos exist in CNF the non-homogeneous case, or can it only be found in the homogeneous case? It is also unclear to us why certain classes in the homogeneous and heterogeneous cases seem to be identical, while other classes have no apparent correlation. We leave these as open questions.

We also performed a cursory analysis of FCA in CNF with other types of fuzzy logic. Once again, we were only able to perform an analysis in the homogeneous case, but doing so demonstrated that certain logics in the homogeneous case produce extremely simple behaviour that does not warrant further study. Simulations in the heterogeneous case proved inconclusive. We cannot say whether these logics in general are of further interest in FCA research.

In general, we did not discover any relationship between FCA in CNF, and their

Boolean counterparts, although that does not preclude the possibility that such a relationship exists. As such, we cannot say whether FCA in conjunctive normal form are meaningful tools to better understand the behaviour of their Boolean counterparts. Nonetheless, we believe that FCA in CNF are interesting, and warrant further study.

Software and Media Licences

The following software libraries and image media have been used. All software libraries and other media have been used under licence, or have been released into the public domain. All other images are my own.

- Java (JDK 6), used under licence, www.java.com
- Janino script evaluator, used under BSD licence, www.janino.net
- GradientSlider, used under Lesser GPL Public Licence (LGPL)
- Substance Look and Feel, used under BSD Licence, https://substance.dev.java.net
- Silk icons, used under Creative Commons Attribution 2.5 licence

Bibliography

[1] Y. Aizawa and I. Nishikawa. Towards the classification of the patterns generated by one-dimensional cellular automata. In G. Ikegami, editor, *Chaotic Dynamical Systems*, pages 210–222, 1986.

[2] K. T. Alligood, T. D. Sauer, and J. A. Yorke. *Chaos: An Introduction to Dynamical Systems*. Springer, 1996.

[3] O. Bandman. Comparative study of cellular-automata diffusion models. In *Parallel Computing Technologies*, volume 1662 of *Lecture Notes in Computer Science*, pages 756–756, 1999.

[4] J. Banks, J. Brooks, G. Cairns, G. Davis, and P. Stacey. The american mathematical monthly. *On Devaney's definition of chaos*, 99(4):332–334, 1992.

[5] H. Betel and P. Flocchini. On the asymptotic behavior of fuzzy cellular automata. *Journal of Cellular Automata*, 6:25–52, 2011.

[6] H. Betel and P. Flocchini. On the relationship between boolean and fuzzy cellular automata. *Theoretical Computer Science*, 412(8-10):703–713, 2011.

[7] H. Betel, P. Flocchini, and A. Karmouch. Convergence and error propagation in self-averaging rules. Manuscript.

[8] G. Braga, G. Cattaneo, P. Flocchini, and C. Q. Vogliotti. Pattern growth in elementary cellular automata. *Theoretical Computer Science*, 145:1–26, 1995.

[9] G. Cattaneo, M. Finelli, and L. Margara. Investigating topological chaos by elementary cellular automata dynamics. *Theoretical Computer Science*, 244:219–241, 2000.

[10] G. Cattaneo, P. Flocchini, G. Mauri, C. Quaranta Vogliotti, and N. Santoro. Cellular automata in fuzzy backgrounds. *Physica D: Nonlinear Phenomena*, 105(1-3):105–120, 1997.

[11] G. Cattaneo, E. Formenti, L. Margara, and G. Mauri. On the dynamical behavior of chaotic cellular automata. *Theoretical Computer Science*, 217:31–51, 1999.

[12] G. Cattaneo and L. Margara. Generalized sub-shifts in elementary cellular automata: The "strange case" of chaotic rule 180. *Theoretical Computer Science*, 201:171–187, 1998.

[13] H. Chaté and P. Manneville. Criticality in cellular automata. *Physica D: Nonlinear Phenomena*, 45:122–135, 1990.

[14] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, 1998.

[15] A. Coxe and C. Reiter. Fuzzy hexagonal automata and snowflakes. *Computers and Graphics*, 27:2003, 2003.

[16] R. Devaney. *A First Course in Chaotic Dynamical Systems: Theory and Experiment*. Addison-Wesley, 1992.

[17] R. Devaney. *An Introduction to Chaotic Dynamical Systems, 2nd Edition*. Westview Press, 2003.

[18] U. D'ortona, D. Salin, M. Cieplak, R. B. Rybka., and J. R. Banavar. Two-color nonlinear boltzmann cellular automata: Surface tension and wetting. *Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 51:3718–3728, April 1995.

[19] P. Flocchini, G. Cattaneo, G. Mauri, C. Q. Vogliotti, and N. Santoro. Cellular automata in fuzzy backgrounds. *Physica D: Nonlinear Phenomena*, 105:105–120, 1997.

[20] P. Flocchini and V. Cezar. Radial view of continuous cellular automata. *Fundamenta Informaticae*, 87(2):165–183, 2008.

[21] M. Gardner. Mathematical games. *Scientific American*, 223:120–123, October 1970.

[22] H. A. Gutowitz. A hierachical classification of cellular automata. *Physica D: Nonlinear Phenomena*, 45:136–156, 1990.

[23] G. A. Hedlund. Endomorphisms and automorphisms of the shift dynamical systems. *Mathematical Systems Theory*, 3(4):320–375, 1969.

[24] W. D. Hillis. The connection machine: A computer architecture based on cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):213 – 228, 1984.

[25] IEEE Standard for Floating-Point Arithmetic. Technical report, The Institute of Electrical and Electronics Engineers, Inc. (IEEE), 2008.

[26] E. Jen. Aperiodicity in one-dimensional cellular automata. *Physica D: Nonlinear Phenomena*, 45:3–18, 1990.

[27] K. Kaneko. Quasiperiodicity in antiferro-like structures and spatial intermittency in coupled logistic lattice. *Progress of Theoretical Physics*, 72, 1984.

[28] K. Kaneko. Spatiotemporal intermittency in coupled map lattice. *Progress of Theoretical Physics*, 74:1033, 1985.

[29] K. Kaneko. Overview of coupled map lattices. *American Institute of Physics: Chaos*, 2:279, 1992.

[30] K. Kaneko. *Theory and Application of Coupled Map Lattices*. Wiley, 1993.

[31] J. Kari. Reversibility and surjectivity problems of cellular automata. *Journal of Computer and System Sciences*, 48:149–182, 1994.

[32] J. Kari. Theory of cellular automata: A survey. *Theoretical Computer Science*, 334(1-3):3–33, 2005.

[33] Y. W. Kwon and S. Hosoglu. Application of lattice boltzmann method, finite element method, and cellular automata and their coupling to wave propagation problems. *Computers and Structures*, 86:663–670, 2008.

[34] H. Y. Lee and Y. Kawahara. On dynamical behaviors of cellular automata. *Bulletin of Informatics and Cybernetics*, 25:21–25, 1992.

[35] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein. Quantum cellular automata. *Nanotechnology*, 4(1):49–57, 1993.

[36] W. Li and N. Packard. Structure of the elementary cellular automata rule space. *Complex Systems*, 4:281–297, 1990.

[37] E. N. Lorenz. Deterministic Nonperiodic Flow. *Journal of Atmospheric Science*, 20(2):130–141, 1963.

[38] S. Maerivoet and B. De Moor. Cellular automata models of road traffic. *Physics Reports*, 419:1–64, 2005.

[39] F. Marchini, M. Meybodi, and V. Soleymani. A Hybrid Method based on Gas Diffusion Model and Fuzzy Cellular Automata for Image Sharpening. *Proceedings of 11th Annual CSI Computer Conference of Iran*, pages 715–720, 2006.

[40] N. Margolus. Cam-8: A computer architecture based on cellular automata. Technical report, MIT Laboratory for Computer Science, 1998.

[41] A. Mingarelli. The global evolution of general fuzzy automata. *J. of Cellular Automata*, 1:141–164, 2006.

[42] A. B. Mingarelli. A study of fuzzy and many-valued logics in cellular automata. *Journal of Cellular Automata*, 1(3):233–252, 2006.

[43] E. F. Moore. Machine models of self-reproduction. *Proceedings of the Symposium on Applied Mathematics*, 14:17–33, 1962.

[44] M. Morse and G. A. Hedlund. Symbolic Dynamics. *American Journal of Mathematics*, 60:815–866, 1938.

[45] J. Myhill. The converse to moore's garden-of-eden theorem. *Proceedings of the American Mathematical Society*, 1963.

[46] N. H. Packard. Adaptation toward the edge of chaos. In *Dynamic Patterns in Complex Systems*, pages 293–301, 1988.

[47] U. Pesavento. An implementation of von neumann's self-reproducing machine. *Artificial Life*, 2:337–354, 1995.

[48] R. Rucker. *Mind Tools: The Five Levels of Mathematical Reality*. Houghton Mifflin Company, 1988.

[49] K. Sutner. Classifying circular cellular automata. *Physica D: Nonlinear Phenomena*, 45:386–395, 1990.

[50] K. Svozil. Constructive chaos by cellular automata and possible sources of an arrow of time. *Physica D*, 45:420–427, 1990.

[51] A. M. Turing. On Computable Numbers, with an application to the Entscheidungsproblem. *Proceedings London Mathematical Society*, 2(42):230–265, 1936.

[52] A. Unkrig. Janino compiler, April 2011. `http://docs.codehaus.org/display/JANINO/`.

[53] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[54] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Springer, 2000.

[55] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, 1983.

[56] S. Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[57] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[58] W. Wright. Open source micropolis (simcity), March 2011. `http://code.google.com/p/micropolis/`.

[59] L. A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[60] K. Zhang, Z. Li, and X. Zhao. Edge detection of images based on fuzzy cellular automata. In *International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, volume 2, pages 289–294, 2007.

[61] K. Zuse. *Rechnender Raum (Computing Space)*. Vieweg, 1969.