On the Effect of Topology on Cellular Automata Rule Spaces

David J. Warne

School of Electrical Engineering and Computer Science

The Queensland University of Technology

A thesis submitted for the degree of

Bachelor of Information Technology (Honours)

December 11, 2013

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signature:

Date:

To my dearest wife, and our little ones;
the two which are here,
and the one that we never met
(The world was not worthy of you).

Acknowledgements

I would like to acknowledge the invaluable input from all of my supervisors, Dr. Ross Hayward, Dr. Neil Kelson, and A/Prof. Dann Mallet. All of your feedback and advice has been very much appreciated. I would especially like to thank Ross for many useful discussions over coffee despite his busy schedule.

This project made use of QUT's high performance compute facility. This facility is administered by QUT's high performance computing (HPC) and research support group. A special thanks to the HPC group for their support, it has been an invaluable contribution to my work.

Abstract

This thesis presents an empirical study of the effects of topology on cellular automata rule spaces. The classical definition of a cellular automaton is restricted to that of a regular lattice, often with periodic boundary conditions. This definition is extended to allow for arbitrary topologies. The dynamics of cellular automata within the triangular tessellation were analysed when transformed to 2-manifolds of topological genus 0, genus 1 and genus 2.

Cellular automata dynamics were analysed from a statistical mechanics perspective. The sample sizes required to obtain accurate entropy calculations were determined by an entropy error analysis which observed the error in the computed entropy against increasing sample sizes. Each cellular automata rule space was sampled repeatedly and the selected cellular automata were simulated over many thousands of trials for each topology. This resulted in an entropy distribution for each rule space.

The computed entropy distributions are indicative of the cellular automata dynamical class distribution. Through the comparison of these dynamical class distributions using the E-statistic, it was identified that such topological changes cause these distributions to alter. This is a significant result which implies that both global structure and local dynamics play a important role in defining long term behaviour of cellular automata.

Contents

List of Figures

List of Tables

Nomenclature

$\mathbf{A}$ A cellular automaton.

$\mathbf{A}_n^N$ A graph cellular automaton with N cells, with graph G topologically equivalent to a surface of genus n.

B The number of bootstrap samples for computing $\Pr(E_n > e)$.

$C^t(x_i)$ The state of cell x_i at time t.

c_α The critical point such that $\Pr(E_n > c_\alpha) = \alpha$.

$C^t(h_i)$ The configuration of the local neighbourhood h_i at time t.

$D(R)$ The distribution of dynamical classes of the rule space R.

E A set of edges $(i; j)$ where $i; j \in V$.

$e(X; Y)$ The two-sample test statistic for the E-test, given random samples X and Y.

E_n The K-sample test statistic for the E-test, given random samples $X_1; \ldots; X_K$ and $n = n_1 + \ldots + n_K$.

f The global configuration transition function of $\mathbf{A}$.

G A connected graph.

g_i The local state transition function for the ith cell of $\mathbf{A}$.

H_0 The *null* hypothesis

h_i The local neighbourhood of cell x_i.

I^t The input entropy.

K The number of independent samples taken for the K-sample test for equality for empirical distributions.

k The size of a local neighbourhood of $\mathbf{A}$

L A regular d-dimensional lattice.

$O(f(n))$ The set of all algorithms with a computational complexity bounded from above by a scalar multiple of $f(n)$.

$\Pr(E_n > e)$ The probability that $E_n > e$ as $n \to \infty$.

R The set of real numbers.

r An offset vector to a neighbour cell in $\mathbf{A}$

R_A The rule space for a cellular automaton of the same form as $\mathbf{A}$.

S The Shannon entropy.

$S^t_{\square}$ The temporal measure entropy.

$S^x_{\square}$ The spatial measure entropy.

U The set of cell neighbourhoods of $\mathbf{A}$.

V A set of vertices of a graph.

W The word entropy.

X The set of objects in a topological space.

x_i The ith cell of $\mathbf{A}$.

Z The set of integers.

Z Wuensche's Z-parameter.

Greek Symbols

$\square$ The statistical significance level, $\square \in (0; 1)$.

Γ The set of local state transition functions of $\mathbf{A}$

$\square$ Langton's $\square$-parameter.

Φ The set of all possible configurations of $\mathbf{A}$.

$\square^t$ The configuration of $\mathbf{A}$ at time t.

Σ The alphabet of $\mathbf{A}$

$\square_q$ The quiescent state.

$\square$ A topology on set X.

$\Theta(f(n))$ The set of all algorithms with a computational complexity bounded from above and below by a scalar multiple of $f(n)$.

$\Omega(f(n))$ The set of all algorithms with a computational complexity bounded from below by a scalar multiple of $f(n)$.

Chapter 1

Introduction

There are many examples of complex phenomena round us. Many complex systems are the result many interacting smaller systems which are themselves simple[1]. Biological systems are built from many interacting simple cells, complex chemical compounds are result of bonds formed between simple atoms, and atoms themselves consist of many sub-atomic particles all interacting at local scales.

Perhaps the most profound example is the human brain which is able to outperform modern computers over a vast range of computational tasks; particularly visual pattern recognition. This performance is achieved despite the brain's relatively low frequency of $\square$ 10^3 neuron firings/second when compared against the frequency of modern micro-processor frequencies which are $\square$ 10^{13} clock pulses/second [5]. The many millions of locally interconnected and coordinated neuron firing patterns in the brain forms a massively parallel, decentralised processor which is capable of such performance.

Cellular Automata, which are mathematical models of interacting simple systems, provide and abstraction from real complex phenomena and enable complexity to be studied as a topic of its own [76]. The research area of mathematics known as *complex systems theory* is concerned with the study of systems in which complexity arises from simple local interactions. The ultimate aim of complex systems theory is to find general laws and principles which can be applied to any real-world example of complexity to gain insight into how the system works [64].

[1] By comparison to the system as a whole.

Some have speculated that the universe itself could even consist of one large cellular automaton [60]. Recently, equivalences have been shown between a cellular automaton and a 1-dimensional quantum field theory [23]. Complex systems theory may provide the insights required to establish the long sought *Grand Unified Theory* [21, 66, 70].

When a cellular automaton is defined on a higher dimensionality[1] the question could be posed, *"What topology should define the space of neighbourhoods?"*. To be in any sort of position to answer this question an understanding of how cellular automaton dynamics can be affected by topology is required. It has been demonstrated that topological perturbations can result in dramatical alterations in the long term evolution of some cellular automata [32, 33, 42]. Surprisingly, very few studies to date have been produced with a focus on the topic of topology and it's ability to control cellular automata dynamics [16]. The question of how topology can affect cellular automata rule spaces is the focus of this study.

This chapter presents an overview of this study. The general background of the research topic is given in Section 1.1 and the specific context of this project is established in Section 1.2. The purpose of the study, the statement of the research questions, and limitations and scope of the study are presented in sections 1.3 and 1.4. The chapter will then conclude with an outline of the structure of the rest of this thesis.

1.1 Background

Cellular Automata can be defined as an array of interconnected *finite state automata*, which are often referred to as *cells* [61]. The next state that a cell will take on is determined by the current state of the cell itself and the of it's neighbours which are typically the immediately adjacent cells. The rules that govern this state transition are provided by a *local state transition function*. All cells within a cellular automaton update synchronously at each time-step according to their local state transition function, which is usually identical across all cells.

[1] Presumably, this would be required for any realistic model of physics.

This synchronised update results in an evolution of the cellular automaton cell state configuration.

Cellular automata are discrete dynamical systems [72]. Across the space of local state transition functions a vast range of dynamical behaviours such as point attractors, oscillators (ordered dynamics), strange attractors (chaotic dynamics), and long complex transients (complex dynamics) are observed [28, 63, 74]. This richness in dynamical possibilities can even be shown to exist in the simplest of cellular automata, the *elementary cellular automata*, which consist of simple binary finite state automata on a one-dimensional array with each cell connected only to the cells to the immediate left and right [62].

These simple dynamical systems have been shown to be powerful enough to represent computational processes [4]. Cellular automata such as the famous *Conway's Game of Life* and *Rule 110* have been shown to be *Turing Complete*, which means they can themselves be considered as computers [11, 60]. The existence of universal computation implies that for any program there exits an initial configuration which encodes the program such that the cellular automaton's evolution represents the program's execution.

There are many cases in which reformulation of a problem as a cellular automaton has aided in gaining new insight into the particular field under study [4]. In general this reformulation step is non-trivial [9], however cellular automaton models have been applied successfully in many fields of Science and Engineering, at times out-performing traditional computing approaches. Contributions to the study of complexity in general will further the ability to model phenomena in a cellular automaton framework which many revolutionise methods of scientific enquiry or provide additional methods for scientists to validate hypotheses [4]. The pursuit of a more in depth understanding of the modelling potential of cellular automata is one of the primary motivations of this study [64].

1.2 Context

The study of complexity using cellular automata usually focuses on the definition of the state transition function. In recent years however, a small number of studies have experimented with the definition to investigate other aspects which

may influence the emergence of complexity from simple interactions [16, 42]. One interesting extension is the removal of the requirement for cells be arranged on a regular lattice [32, 55]. In the most general case, cellular automata can be defined on graphs. These *graph cellular automata* (also called *generalised automata networks* [55]) can be defined on any connected graph.

For some graph cellular automata it has been shown that graph rewiring can completely change the dynamics, without changing the definition of the local state transition function [16, 32]. It was also show that some graph types had a higher concentration of complex cellular automata [33]. By the introduction of random wirings in an otherwise regular lattice, a complex behaviour may completely degrade to a simple fixed pattern [16, 42]. Subtle variations in the dynamics of a complex cellular automaton defined on a sphere have also been shown to expose novel logic implementations [57].

In general though, very little research has been done on the topic of topology and its influence on cellular automata dynamics [16]. This is surprising, as it has many applications for improving the performance and/or stability of systems [42, 55].

1.3 Purpose

The purpose of this study is to contribute to the knowledge base of complex systems theory through the investigation of how cellular automata behaviour can be influenced by topology. Most of the previous studies in this area have looked at random re-wirings, which completely break the homogeneity of the local neighbourhood, which is not necessarily desirable. In this study, topological variations which preserve homogeneity of local neighbourhoods were investigated instead.

This thesis presents a quantitative study of the variations in cellular automata observed under topological variations that preserve graph regularity. A graph is *regular* when all graph vertices have the same *degree*, that is, they have the same number of incident edges and hence neighbours. This constraint on topology allows any cellular automaton rule to be defined consistently across any topology, thus enabling us to study the global structure of the most complete rule space.

This is one of the major restrictions in previous studies, in which only a very small subset of rule types can actually be defined consistently across topologies with varying neighbourhood sizes.

Specifically, topologies which are isomorph to some mesh representing a 2-manifold were selected. The variations in topology were based on the topological *genus* of the mesh. Thus our study can be considered as a broad quantitative study similar to that of Marr et al. [32, 33] but looking at topologies more in line with those considered by Ventrella [57].

Dramatic changes in cellular automata dynamics are referred to as *critical phenomena* [16]. The research questions of this study a centered around the critical phenomena caused by topological variations of this kind. Specifically this study will aim toward answering the following questions:

- Can critical phenomena be caused by topological variations which preserve homogeneity of local neighbourhoods?
- Do critical phenomena occur often enough for to cause a significant change in the distribution of cellular automata dynamical classes across the rule space? If so, which topologies are best for maximising the concentration of a particular class.
- Can other dynamical differences still be observed across topologies when no critical phenomena are observed?

In this study, the dynamical classes of thousands of cellular automata are inferred using a statistical mechanics approach in which a measure of dis-order is computed through repeated simulation from random initial conditions. This allows a dynamical class distribution to be constructed for rule spaces on each topology, which is then compared using a distribution-free statistical test for equality.

1.4 Limitations and Scope

The scope of this project is restricted to two dimensional, binary-state cellular automata in the triangular tessellation (i.e., a triangular grid) [7, 60]. This has

been done primarily to reduce the rule space size so that a statistical mechanics approach is feasible. The triangular tessellation is sufficient to represent any surface, which reduces the complexity of topology construction. Very few studies have investigate the triangular tessellation [7], rather the square tessellation is the most common two dimensional representation [57, 60]. Binary-state cellular automata in the triangular tessellation can be constructed with a neighbourhood size of 4, which is only one greater than the one dimensional elementary cellular automata; thus these simple triangular neighbourhoods may be considered to be the two dimensional equivalent of elementary cellular automata.

This study applies an experimental approach; thus evidence for or against a given hypothesis can be provided, but it cannot generally be proven or dis-proven via this approach. Such proof would require a detailed formal mathematical approach, with is far beyond the scope of this study. This study, however, will provide a basis for which a more mathematically formal proof may be constructed for the necessary and sufficient conditions for the existence of critical phenomena in graph cellular automata.

The neighbourhood homogeneity preserving topologies considered in this study are those which are derived from triangular meshes of a sphere, torus, and double-torus. It follows that any trends observed in this study are restricted to these topologies. However, conjecture is still made at times that such trends extend to other topologies with genus $\mathsf{g} > 2$.

1.5 Thesis Outline

In Chapter 2, a detailed review of the literature relating to cellular automata and critical phenomena is presented. This chapter includes relevant formal definitions of cellular automata and geometric topology, discusses dynamical classification schemes, and summarises current methods of analysis. The few studies to date related to critical phenomena caused by topological variations are also given an in depth exposition. Thus highlighting the current state of research and how this study represents a unique contribution.

Chapter 3 describes the research methodology applied in this study. This includes derivation of the *null* hypothesis and the experimental design. Technical

details on cellular automata and topology generation, entropy error analysis, simulation methods, and data analysis algorithms are all presented in this chapter.

In Chapter 4 the results and discussion is presented. The results section describes the direct results of error analysis tests, simulations, and statistical hypothesis test with little or no interpretation. The discussion section, provides more of an interpretation of the results and provides a basis for some conjecture.

Finally, Chapter 5 concludes the thesis. This chapter will relate findings back to the research questions, and highlight future work that may be done to validate and extend this work.

Chapter 2

Literature Review

In this chapter, I will present a review of the literature from the study of cellular automata. Due to the long[1]history of the study of cellular automata and, more broadly, discrete dynamical systems, it would be an impossible task to review all the literature. The sections that follow provide details of key topics from the literature that are foundational to my study such as the definition of cellular automata, the analysis of cellular automata evolution, and the study of critical phenomena induced by topological variations.

2.1 Cellular Automata

A *cellular automaton* is commonly described as a lattice of identical *finite state automata* (i.e., simple machines that can only be in a finite number of states). These finite state automata, typically referred to as *sites* or *cells*, update their states synchronously with each other [4, 61]. The next state of each cell is determined from it's current state and the states the cells in it's local neighbourhood (i.e., adjacent cells in the lattice). The sequence of all cell states forms the cellular automaton's *configuration*. A new configuration is determined from the synchronous update of the cells, resulting is a sequence of configurations referred to as the automaton's *evolution*.

[1]That is, long in terms of the age of Computer Science.

Cellular automata can be considered as *discrete dynamical systems*; the discrete version of *continuous dynamical systems* such as partial differential equations. Despite the simplicity of their construction, their evolutions have been shown to exhibit a wide range of dynamical characteristics. This ranges from simple convergent behaviour to complex behaviour capable of mimicking real phenomena or even performing computations [11, 28, 76].

The range of dynamics possible even from simple cellular automata makes them a powerful tool for modelling phenomena, with applications in physics, theoretical biology, engineering and computer science. The study of cellular automata has lead to the field of *complex systems theory*; a field of mathematics dealing with how complex global dynamics can emerge from simple interacting structures [64]. It may be that complex systems theory holds the key to understanding the fundamental properties of "self-organising" and "self-replicating" systems, the nature of computation, and to develop a *grand unified theory* of the universe [21, 23, 70].

2.1.1 History of Cellular Automata

The first formalisation of cellular automata is attributed to John von Neumann though his study of self-replication in the late 1940's to 1950's [59]. By using cellular automata Von Neumann was able to build a universal constructor, and hence a machine that could build a copy of itself.

The study of cellular automata progressed slowly though the 1960's and 1970's, although through this period some important discoveries were made. John Conway constructed a 2-dimensional binary cellular automaton now known as *Conway's game of life*, which was shown to be capable of supporting complex patterns [18]. Conway's game of life was proven to be computationally universal by Robert Wainwright in 1974 [60]. During this time Stuart Kauffman also developed is variant of cellular automata called *random boolean networks* which he showed to be an accurate model predicting the number of cell species derived from a genetic regulatory network of a given size [26].

An explosion of interest in cellular automata began in the 1980's when Stephan Wolfram present a detailed study of the statistical mechanics of the simplest class

of cellular automata [61, 62, 63]. Wolfram showed that even these *elementary cellular automata* could support very complex behaviour [63]. Wolfram suggested that cellular automata were ideal for modelling systems that are difficult to model with traditional continuous mechanics [64]. Wolfram also coined the term *complex systems theory* referring to the study of complexity in an abstract context [64], and posed twenty unsolved problems of interest in the field [65]. Complex systems theory was shown to have many implications for the study of real complex phenomena [64, 66].

In 2002 Wolfram release his book titled *A New Kind of Science* which is dedicated to presenting findings in the study of cellular automata as profoundly important to all areas of science [70]. Soon after the publication of *A New Kind of Science*, Matthew Cook presented his proof that the elementary cellular automaton *Rule 110* was computationally universal [11]. This proof supported conjectures made by Wolfram.

2.1.2 Formal Definition of Cellular Automata

There are many differences in notation and terminology between authors when referring to cellular automata [4, 28, 32, 55, 61]. In this section, the common features from the literature are extracted to present a formal definition of cellular automata. This formal definition will be utilised heavily in the proceeding chapters.

2.1.2.1 Regular Cellular Automata

Usually when the term *cellular automata* is used in the literature it is referring to *regular cellular automata* [32, 55, 57]. A regular cellular automaton as a 4-tuple consisting of a d-dimensional lattice (which can be infinite), a finite alphabet of states, a set of offsets representing the neighbourhood and a local state transition function. More formally we define a cellular automaton as in Definition 1.

Definition 1 *Let* $\mathbf{A} = (\mathsf{L}; \Sigma; \mathsf{U}; \mathsf{g})$ *define a* d*-dimensional cellular automaton where* $\mathsf{L} \square \mathsf{Z}^{\mathsf{d}}$ *is a set of* d*-tuples representing a regular lattice of cells and* x_{i} 2 L *is the* i*th cell location,* Σ *is a finite set of states referred to as the alphabet,*

$U = f h_i : x_i \ 2 \ L g$ *is a set of local neighbourhoods* $h_i = f y : y = x_i + z \ ^ \ (y \ 2 \ L) \ ^ \ (z \ 2 \ f \ \Box r; \Box r + 1; : : : ; 0; : : : ; r \ \Box \ 1; r g^d) g$ *of uniform size* k*, and* $g : \Sigma^k \ ! \ \ \Sigma$ *is the local state transition function.*

At any given time t each cell is associated with a single state $\Box \ 2 \ \Sigma$.

Definition 2 *Let* $C : L \ ! \ \ \Sigma$ *be a mapping from a cell* $x_i \ 2 \ L$ *to a state* $\Box \ 2 \ \Sigma$ *such that* $C^t(x_i)$ *represents the state of cell* x_i *at time* t*. Let* $C^t(h_i)$ *denote the local neighbourhood configuration* $f C^t(y) : y \ 2 \ h_i g$.

Definition 2 leads to the construction of the global configuration of a cellular automaton.

Definition 3 *Let* $\Box^t = f C^t(x_i) : x_i \ 2 \ L g$ *be the configuration of a cellular automaton* **A** *at time* t*.* $\Box^t \ 2 \ \Phi$ *where* Φ *is the set of all possible configurations of* **A**.

The evolution of a cellular automaton is the sequence of configurations generated by repeated synchronous application of the local state transition function g. This can be defined as a recurrence relation in terms of a global configuration transition function.

Definition 4 *Let the recurrence relation* $\Box^{t+1} = f (\Box^t); \ t \ \Box \ 0$ *be the evolution of* **A***, where* $f \ : \Phi \ ! \ \ \Phi$ *is the global configuration transition function* $f = f (\Box^t; \Box^{t+1}) : \Box^t = f C^t(x_i) : x_i \ 2 \ L g \ ^ \ \Box^{t+1} = f g(C^t(h_i)) : h_i \ 2 \ U g g$.

2.1.2.2 Random Boolean Networks

Random boolean networks are discrete dynamical systems similar to regular *binary* cellular automata, with two critical differences. Firstly, the local neighbourhood constructed via an arbitrary wiring (usually random) [26]. Secondly, the local update rule for each cell is also arbitrary (usually random with some constraints) [26, 71].

Formally we have,

Definition 5 *Let* $\mathbf{A} = (L; U; \Gamma)$ *be a random boolean network where* $L \subseteq \mathbb{Z}$ *is a set of cells,* $U = \{h_i : x_i \in L\}$ *is the set of randomly constructed neighbourhoods* $h_i = \{x_j : x_j \in L \wedge \forall j \in [1; |L|]; (\Pr(J = j) = 1/|L|)\}$ *with* $\forall i; |h_i| = k$*, and* $\Gamma = \{g_i : x_i \in L\}$ *is the set of local state transition functions* $g_i : \{0; 1\}^k \to \{0; 1\}$.

Once the neighbourhoods h_i and the state transition functions g_i have been constructed they remain fixed and deterministic throughout the systems evolution. Though the definition could easily be extended to a n-ary alphabet, random boolean networks were specifically developed to study genetic regulatory networks from a theoretic biology point of view [26, 71, 73, 78]; That is, a gene is suppressed or expressed; which is naturally binary.

2.1.2.3 Graph Cellular Automata

Regular cellular automata and random boolean networks can be further generalised to cellular automata defined on any connected graph [33, 55]. These *graph cellular automata* (also referred to as *generalised automata networks* [55] or *cellular automata on graphs* [32, 33]) Building on the formalism of regular cellular automata in Section 2.1.2, a graph cellular automaton is constructed by replacing the regular lattice L for a connected graph G.

Definition 6 *Let* $\mathbf{A} = (G; \Sigma; U; \Gamma)$ *define a graph cellular automaton where* $G = (V; E)$ *is a connected graph with vertices* $V \subseteq \mathbb{Z}$ *and edges* $E \subseteq V \sqsubset V$*,* Σ *is a finite set of symbols referred to as the alphabet,* $U = \{h_i : i \in V\}$ *is the set of neighbourhoods* $h_i = \{j : (i; j) \in E \lor (j; i) \in E\}$*, and* $\Gamma = \{g_i : i \in V\}$ *is the set of local state transitions functions* $g_i : \Sigma^{|h_i|} \to \Sigma$.

Note that Definition 6 is similar in many respects to Definition 1. The cells of the automaton $\mathbf{A}$ are represented by the vertices of the graph G. The neighbourhoods $h_i \in U$ consist of all cells connected via the set of edges E[1]. Definition 6 assumes G is a undirected graph, but this is not a requirement and may be easily changed by re-defining $h_i = \{j : (j; i) \in E\}$.

[1] Typically a cell is considered connected to itself (i.e., $\forall i \in V; \exists (i; i) \in E$), however this is not a requirement.

Table 2.1: Example of the Wolfram code naming convention for the elementary cellular automaton *Rule 110*. Here $_n$ is the nth possible neighbourhood configuration, g is the local state transition function, s is the number of states.

$_n$	000	001	010	011	100	101	110	111
$g(\ (_n))$	0	1	1	1	0	1	1	0
$s^n g(\ _n)$	$2^0 \square 0$	$2^1 \square 1$	$2^2 \square 1$	$2^3 \square 1$	$2^4 \square 0$	$2^5 \square 1$	$2^6 \square 1$	$2^7 \square 0$
$C_g = 110 =$	0 +	2 +	4 +	8 +	0 +	32 +	64 +	0

2.1.2.4 Other Definitions

A few other other constructs relating to cellular automata appear often in the literature. These include common naming conventions, state types, and rule types. The following definitions are presented: The *Wolfram code*, the *quiescent* state, and totalistic/outer-totalistic rules.

A common naming convention in the form of a numerical code is applied to cellular automata, this is often referred to the *Wolfram code* (after Stephan Wolfram who first utilised such a scheme [62]). A cellular automaton's Wolfram code $C_g$is generated from it's local state transition function g,

$$C_g = \sum_{n=0}^{s^k \square 1} s^n g(\ _n) \tag{2.1}$$

where $s = |\Sigma|$, $k = |U|$, and $_n \in \Sigma^k$ represents the nth possible neighbourhood configuration. An example of Equation (2.1) applied to a binary cellular automaton[1] is provided in Table 2.1. It is worth noting that the Wolfram code only applies to regular cellular automata or graph cellular automata with homogeneous state transition functions.

Typically a specific state is defined called the *quiescent* state. This state is denoted as $\square_q \in \Sigma$ and is defined as the state such that $g(\square_q; \square\square; \square_q) = \square_q$ [28]. Conceptually, the quiescent state is considered the "dead" state, and dead state in a dead state remains dead.

[1] Note that the result is simply the decimal representation of the binary number encoded the outputs of g.

Two special classes of cellular automata rules are of particular interest; namely *totalistic*, and *outer-totalistic* rules [7, 32, 33, 62]. These are of interest because results are independent of orientation. Rules from these classes can be represented more simply than a lookup table which is the most general form of the local state transition function[1].

A output of a *totalistic* rule is only dependent on the sum of the non-quiescent states within the cell's local neighbourhood. That is, $g = f(y)$, where $y : \Sigma^k \; ! \;\; Z$ and $f \; : Z \; ! \;\; \Sigma$. The function y is given by

$$y(C^t(h_i)) = \sum_{x2jh_ij} R(C^t(x)) \tag{2.2}$$

where $R : \Sigma \; ! \;\; Z$ is a function which maps states to numerical values, with the constraint $R(\square_q) = 0$.

Outer-totalistic rules (also called *semi-totalistic* rules [7]) can be expressed as a function dependent on the sum of the states in a cell's *outer*-neighbourhood (i.e., the local neighbourhood excluding the cell itself) and the cell's current state. That is, $g = f(y; \square)$, where y is defined as in Equation (2.2), and $f \; : Z \square \Sigma \; ! \;\; \Sigma$.

The sequence of configurations generated by a cellular automaton's evolution forms a space-time pattern. This pattern is the object of interest in the study of cellular automata dynamics. The space-time pattern is also useful when visualising cellular automata behaviour.

2.2 Dynamical Characteristics of Cellular Automata

Cellular automata are capable of supporting a wide range of dynamical characteristics [61, 62, 63]. Broadly, cellular automata fall into three dynamical classes: *ordered*,*chaotic*, and *complex* [27, 28, 32, 76]. These are characterised by the expected behaviour of a cellular automaton in the long time evolution.

Ordered cellular automata will tend to ultimately converge to a single fixed configuration, akin to a point attractor in continuous dynamics. Chaotic cellular

[1] Lookup table representations can become unwieldy when the number of possible neighbourhoods is large.

automata are highly sensitive to initial configurations and the long time behaviour will typically be almost "random"[1], akin to a strange attractor in continuous dynamics. Complex cellular automata have unpredictable long time behaviours, possibly converging to a single configuration or forming complex oscillatory cycles. Complex cellular automata may even generate "life-like" gliders, supporting self-replication.

In the early to mid-1980's, Stephan Wolfram performed an extensive study on the simplest class of cellular automata,called *elementary cellular automata*, which are binary (i.e., $\Sigma = \{0; 1\}$) 1-dimensional automata [61, 62, 63]. Wolfram showed that, even with this minimalistic definition, cellular automata are capable of supporting the full spectrum of dynamical classes. Wolfram characterised what he observed into four qualitative dynamical classes [63]. Wolfram's four classes were originally presented as follows:

- Class I: Evolution leads to a homogeneous (i.e., all cells are the same state) configuration.
- Class II: Evolution leads to a set of simple separated or periodic structures.
- Class III: Evolution leads to a chaotic pattern.
- Class IV: Evolution leads to complex localised patterns, sometimes long-lived.

Examples are shown in Figure 2.1

In terms of the broad dynamical classes defined in the preceding paragraph, Class I and Class II cellular automata are ordered, Class III cellular automata are chaotic, and Class IV cellular automata are complex [76]. Wolfram's classification scheme has been adopted by many, however it suffers from being informal and subjective [14]. As a result, others have attempted to define a more formal classification [27, 28, 39, 41].

Wolfram and others theorised that Class IV cellular automata are capable of universal computation, even for elementary cellular automata [28, 63]. That is,

[1] The long term behaviour is not actually random, but is still completely deterministic. However, the information entropy of the space-time pattern will be close to 1

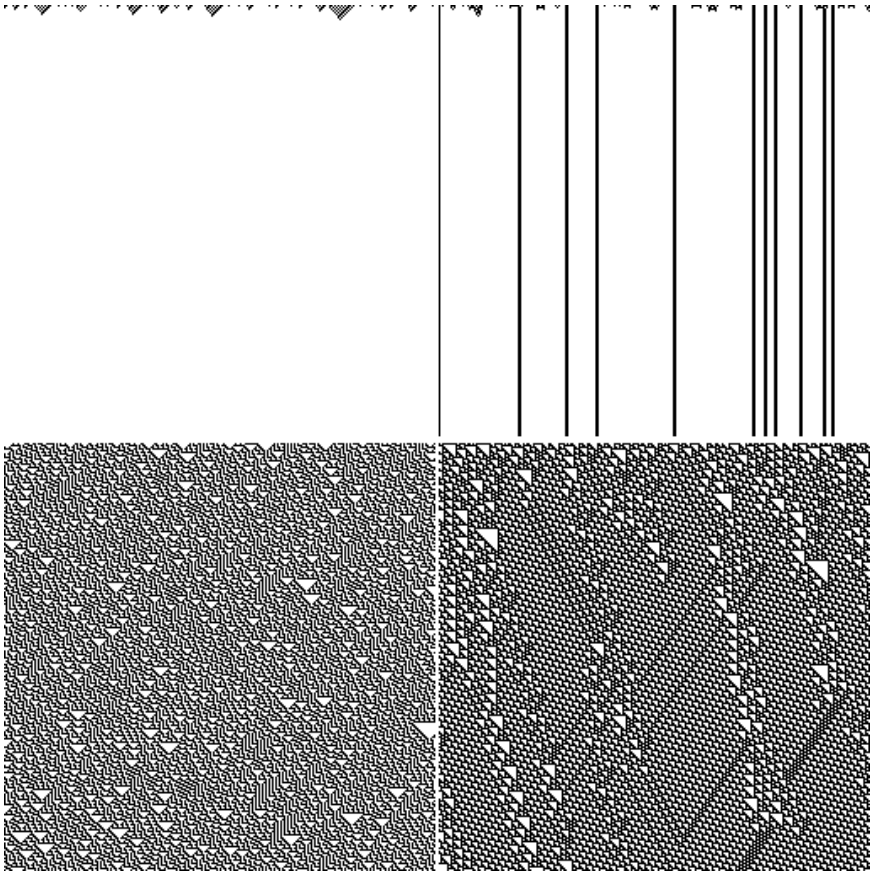

Figure 2.1: Examples of Wolfram Classes. Clockwise from top left: Class I (homogeneous), Class II (periodic), Class III (chaotic), Class IV (complex).

Class IV cellular automata may be considered as computers in their own right, capable of computing any finite computation. Wainwright had already proven that the binary two-dimensional cellular automaton, *Conway's Game of Life* [18] (see Figure 2.2), to be *Turing-complete* (i.e., Capable of simulating any universal Turing-Machine [56]) in the mid-1970's [60]. Cook further validated Wolfram's theory by proving that the elementary cellular automaton, *rule 110* (see Figure 2.2), to be Turing-complete [11, 12].

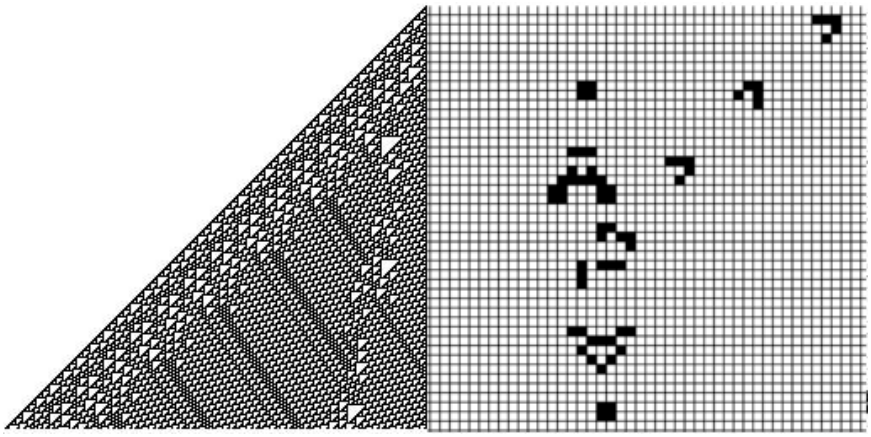

Figure 2.2: Turing Complete Cellular Automata. The left image shows the time-space pattern formed by *Rule 110* given a simple 1D initial configuration. The right image shows a single time-step of the 2-dimensional cellular automaton *Conway's game of life.*

A formally undecidable theorem has no finite formal proof to determine it's truth or falsehood. Gödel's Theorem famously showed that formal undecidability is an inevitable consequence of mathematical completeness [19, 22]. In the general case, the long time behaviour of a universal Turing machine is formally undecidable, shown by Turing's diagonal argument known as *Halting Problem*[1]. The existence of Turing-complete cellular automata has profound implications for what can be determined, either analytically or computationally, about their long time evolution in general. It leads to the inevitable conclusion that the simplest

[1] The Halting Problem is effectively the computational view of Gödel's Theorem.

simulation of a universal cellular automaton is the automaton itself [4, 11, 66, 69]. This is a property referred to by Wolfram as *computational irreducibility* [66]. As a result of computational irreducibility in cellular automata, there does not exist a general method to directly compute the dynamical class of a cellular automaton.

2.3 The Quantitative Analysis of Cellular Automata

As there is no general method to compute the dynamical class of any cellular automaton directly, in practice, various metrics based on either the form of the local state transition function [28, 74], the global configuration transition graph [26, 68, 72, 78], or analysis of the space-time pattern from a statistical mechanics point of view [32, 33, 61, 74]. These methods are not mutually exclusive, and may even compliment each other. In this section, common methods from the literature are discussed.

2.3.1 Rule Table Analysis

The most general representation of a cellular automaton's local state transition function is a look-up table [28, 71]. Analysis of the rule look-up table can be used as a rough indicator of the cellular automaton's behaviour. This approach has the advantage of computational efficiency as the cellular automaton's evolution is never simulated.

The most famous of the rule table based methods is Langton's □-parameter [28]. The □-parameter is defined as the probability that a cell transitions to a non-quiescent state (as defined in Section 2.1.2.4). Using the formalism in Sections 2.1.2 to 2.1.2.4, and given n possible entries in g that map to the quiescent state $□_q$, then

$$□ = \frac{k^N \; □ \; n}{k^N} \tag{2.3}$$

where $k = |\Sigma|$ and $N = |h_i|$.

Initial analysis of Equation (2.3) reveals that $□ = 0{:}0$ when $n = k^N$ indicating that all possible state transitions map to $□_q$; this represent maximal homogeneity.

When all states are equally likely □ = 1 □ $\frac{1}{k}$, which occurs when $n = k^{N} \square 1$; this represents maximal chaotic behaviour. Langton showed that cellular automata with □ closer to the homogeneous extreme tend to be from Wolfram Classes I and II, whereas □ closer to the chaotic extreme tend to be from Wolfram Classes III [28].

Langton conjectured that as □ is varied from □ = 0:0 to □ = 1 □ $\frac{1}{k}$ it will eventually cross a critical point in which a phase transition occurs from order to chaos. This phase transition is referred to as the *edge of chaos*; Langton supposed that Wolfram class IV cellular automata would tend to occur at this point. Langton's experiments are supported by some other studies [35, 72] but contested by others [34].

Another rule table method is known as Wuensche's Z-parameter [74]. This parameter is computed as a side effect of Wuensche's *reverse algorithm* [78], which is a method for directly solving the so-called *Eden problem* [3, 6, 24, 41, 49, 50]. The Eden problem seeks to resolve the question, for a given configuration $\square^{t}$ "*Does there exist a pre-image* $\square^{t \square 1}$ *such that* $\square^{t} = f(\square^{t \square 1})$?"[1].

Wuensche's reverse algorithm attempts to construct a pre-image by taking an initial guess at a cells state, followed by deductions based on the rule look-up table. The next cell is said to be found deterministically if only one possibility exists, based on the cells determined (and assumed) to that point.

For an elementary cellular automaton Wuensche's reverse algorithm proceeds as follows. Assume the first k □ 1 cells in the pre-image are known $p_1; :::; p_{k \square 1}$, then there are two entries in the rule look-up table which are potential pre-images ,say $\square_1$ and $\square_2$, for the k=2-th cell in the original configurations. Let the lookup table entry of $\square_1$ and $\square_2$ be denoted by $!_1$ and $!_2$ and let $\square_{k=2}$ denote the state of cell k=2 in the original configuration. If $!_1 = !_2 \neq \square_{k=2}$ then the previous k □ 1 states are not part of a valid pre-image. If $!_1 = !_2 = \square_{k=2}$ then both $\square_1$ and $\square_2$ could be part of a valid pre-image and both need further processing. However if $!_1 \neq !_2$ then one only them must be equal to $\square_{k=2}$ that is the k cell state is uniquely and deterministically known.

[1] The reader is referred to Appendix A for details on the Eden problem and some work on the problem performed in early stages of this project.

The Z-parameter is defined as the probability that the next unknown cell state can be found deterministically. The range of the Z-parameter is Z 2 [0; 1]. Z = 0 indicates high convergence and order, and Z = 1 indicates low convergence and chaos [74, 78]. Wuensche compares his Z-parameter against Langton's □-parameter and indicated that they are similar in nature [74], unfortunately he does not provide an argument for the advantage of using the Z-parameter over the □-parameter. Wuensche discovered that the Edge of chaos tends to be around Z 2 [0:5; 0:75],that is, Wolfram Class IV cellular automata are expected to occur in this interval.

2.3.2 Configuration Transition Graph Analysis

The configuration transition graph is a representation of the global dynamics of a cellular automaton [62, 77], similar to a phase portrait from continuous dynamics [78]. Each possible configuration of a given cellular automaton is represented as a graph node,and each configuration transition is represented as a directed edge [66, 68, 72].

For finite cellular automata, these graphs will always form *attractor cycles*, which are cycles for which every connected configuration in the graph will converge to. Leaf nodes represent configurations without a pre-image, also called *Garden of Eden* configurations. The set of configurations which form a chain from aleaf node to an attractor cycle are called *transients*.

Properties of configuration transition graphs can provide good indicators of cellular automata behaviour. Useful properties include the mean attractor cycle length, the number of cycles, the mean transient length, the number of leaf nodes, and average node in-degree [26, 68, 72, 76].

The most significant problem with direct analysis of the configuration transition graph is it's computational overhead. The number of nodes in the graph will grow exponentially with the number of cells, making analysis of cellular automata with large cell counts computationally intractable [77]. Of course an alternate approach is to simply construct sub-sections of the graph based on a random sampling.

2.3.3 Statistical Mechanics

The study of cellular automata dynamics can be viewed from a statistical mechanics perspective. Statistical mechanics deals with the study of the macroscopic behaviour of a system in terms of is microscopic components based on the probabilities of micro-states. A cellular automaton is considered as a statistical ensemble of cells in which a stochastic process drives cell state changes [62].

The most fundamental concept here is that of *entropy*, which is defined as the amount of disorder in the system. Entropy can also be considered as the amount of unknown *information* about the systems state. For example, a Class I cellular automaton will have minimal entropy as all information about its ultimate state is known after very few time-steps.

Shannon's information formula relates the entropy of a system to the probabilities of each possible state the system can be in [43]. The Shannon entropy of a cell x_i is given by,

$$S_i = -\sum_{j=0}^{k-1} p_j^i \log_s p_j^i \tag{2.4}$$

where $s = |\Sigma|$ and $p_i(\sigma_j)$ is the probability that cell x_i is in state σ_j, that is, $p_j^i = \Pr(C^t(x_i) = \sigma_j)$. The average Shannon entropy $S = \frac{1}{N}\sum_{i=0}^{i=N-1} S_i$ can be used to distinguish stationary patterns from those that are chaotic or oscillatory [32, 33, 66]. This can be understood since a single state will dominate in any stationary pattern which will yield $S \approx 0$; a chaotic pattern will tend to have near equal probabilities for all states yielding maximal entropy $S \approx 1$.

The micro-states of the system need not relate to a single cell's state as it is in Equation (2.4), sequences or blocks of cells may be of interest. Wolfram applied such a method which he referred to as the *spatial measure entropy* [66],

$$S_\sigma^x(X) = -\frac{1}{X}\sum_{j=0}^{s^X-1} p_j^x \log_s p_j^x \tag{2.5}$$

where p_j^x is the probability of the j th possible spatial block of length X occurring, given s^X possibilities. Wolfram also defined a *temporal measure entropy*,

$$S_\sigma^t(T) = -\frac{1}{T}\sum_{j=0}^{s^T-1} p_j^t \log_s p_j^t \tag{2.6}$$

where p_j^t is the probability of the j th possible temporal sequence of length T. By using spatial/temporal block configurations as the micro-states of the system a more detailed picture of the dynamics of the system can be constructed. This is due to correlations between cells being taken into account, as opposed to completely independent of each other as in Equation (2.4).

Unfortunately the spatial measure entropy is difficult to define for graph cellular automata, so more emphasis is placed on the temporal information. Marr and Hütt applied what they refer to as the *word entropy* [32, 33] which is given by,

$$W_i = \sum_{l=1}^{X^T} p_l^j \log_T \left[p_l^j \right] \tag{2.7}$$

where p_l^j is the probability of a contiguous sequence of length l (i.e., an l-word). Note that Equation (2.7) is very similar to Equation (2.6) except sequences of any length are considered and only contiguous sequences are of interest. The average word entropy $W = \frac{1}{N} \sum_{i=0}^{i=N-1} W_i$ can be plotted against the average Shannon entropy to obtain a visualisation of a cellular automata rule space. Marr and Hütt show that this visualisation is roughly segmented into regions relating to the Wolfram Classes [32].

Wuensche applies a measure he refers to as the *input-entropy* [71, 77, 78]. The input entropy measure is based on the probabilities of the occurrences of local neighbourhood configurations. Wuensche defines the input entropy I and time t as

$$I^t = \left[\frac{1}{N} \sum_{j=0}^{s^k-1} Q_j^t \log_{s^k} \left[\frac{Q_j^t}{N} \right] \right] \tag{2.8}$$

where $\delta i; k = j h_i j$, and Q_j^t is the frequency of the j th possible local neighbourhood configuration out of the possible s^k at a given time t. Note that Wuensche's input entropy (Equation (2.8)) is effectively Wolfram's spatial measure entropy (Equation (2.5))with $X = k$.

By using summary statistics on the time series of the input entropy, Wuensche was able to create a near-automatic classification scheme for cellular automata which is in close agreement with Wolfram's Qualitative Classes [74]. Class I and Class II cellular automata will have a low input entropy mean and variance; Class

III will have a high input entropy mean, but low input entropy entropy variance; Class IV will tend to have a medium/high input entropy mean and variance [76].

2.4 Cellular Automata in Applications

The abstract nature of the study of cellular automata dynamics make cellular automata a powerful tool for the study of real world complex systems, in without needing to resort to traditional continuous dynamical systems using partial differential equations [64, 70]. Domains of applications include biology, physics, engineering, and computer science [4]

Theoretical biology has many applications for cellular automata and related discrete dynamical systems. This is largely due to common characteristics shared by real biology systems and cellular automata, such as global behaviour defined through local interacting structures. From the very beginning of the field, cellular automata have been studied as abstractions of living systems, or *Artificial Life* [29]. In this context cellular automata have been used to investigate phenomena such as self-reproduction [1, 20, 59],self-organisation [25, 62, 75], and adaptation via mutation [8, 26, 28, 35] in a general form. Cellular automata have also been applied to model real biological cell interactions. Examples include models of the tumor-immune system interactions [30], and of genetic regulatory networks [26, 73].

In physics there are also no shortage of applications. Some cellular automata can mimic a continuum system [67] resulting in direct applications for computational thermodynamics and hydrodynamics [40]. Cellular automata models of phenomena such as complex turbulent fluid flows [65] and flows though porous media with results of the same accuracy as obtain through discretisations of the Navier-Stokes equations [47]. In a theoretical setting, the discovery of computationally irreducible discrete dynamical systems leads to the conclusion that many unsolved problems of theoretical physics may in-fact be computationally irreducible [66].

Other physical sciences are not without cellular automata applications. Growth of crystalline structures such as snowflakes can be modelled with relative ease with

cellular automata [65]. Cellular automata are used in the study of chemical processes such as reaction-diffusion [1]. Geomorphological models of landslides with very accurate prediction capabilities [9]. Even civil engineering applications such as storm water network design have been successfully applied [2].

The Turing-completeness of cellular automata makes them of particular interest to computer science. Cellular automata have many useful computational properties that a traditional Von Neumann architecture does not, such as decentralised processing, massive parallelism, fault tolerance, and simplicity [5, 69]. Computer architectures have certainly been influenced by cellular automata [4]. An example of a special purpose computer architecture was the CAM (Cellular Automata Machine) series architecture developed at the *MIT Laboratory for Computer Science* [31, 53, 54]; this architecture was designed specifically for simulation of cellular automata and was thousands of times faster that Von Neumann based CPUs of the day [53]. Other Cellular automata based architectures have also been proposed [10], and implemented for special tasks [58]. A cellular automata processor has even been realised in an organic mono-layer [5], a form of processor which could revolutionise computer systems. Other applications from computer science include image processing [38], encryption [64, 76], and integrated circuit manufacturing [45].

2.5 Geometric Topology

Topology is a field of mathematics which studies the properties of *topological spaces* which are invariant under certain transformations [44]. *Geometry topology* is a subfield of topology in which topological spaces define n-dimensional manifolds. Essentially my project involves comparing the dynamics of cellular automata defined of different topological classes of 2-dimensional manifolds embedded in 3-dimensional space (i.e., surfaces).

Informally, a topological space can be defined as a set of "points" X, and a topology on X. A topology on X is a set $\square$ of subsets of X. The elements of $\square$ define the "closeness" of elements in X. A topological space need not be

"geometric", but for my project the set X is always a set of points and τ is effectively the set of neighbourhoods[1].

Two properties of a topological space that are straight forward to describe are *connectedness* and *genus*. A topological space is connected if a path can be drawn from any point to any point. Geometrically this means a connected topology is a single piece, as shown in Figure 2.3.

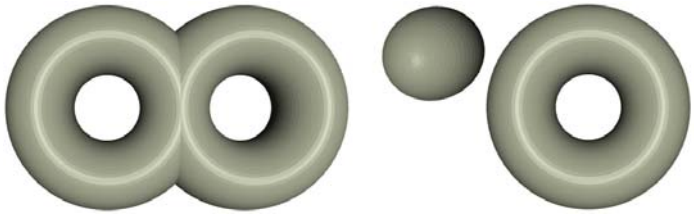

Figure 2.3: Example of topological connectedness. The "eight-like" object on the left is connected, however the object on the right consisting of a torus and a sphere is not connected

The genus of a topological space is the maximum number of times you can remove the points included in a closed curve from X and result in a connected topological space. A geometric example is shown in Figure 2.4. For closed surfaces the genus is equal to the number of "holes" it has. A sphere is a genus 0 surface, and a torus is a genus 1 surface.

Figure 2.4: Topological genera. From the left: Genus 0 (sphere), Genus 1 (torus), Genus 2 (double torus)

[1] τ is really the set of neighbourhoods, neighbourhoods of neighbourhoods, etc...

Two topological spaces are equivalent if the first may be *continuously deformed* into the second. Two surfaces with the same genus are topologically equivalent, even though geometrically they may differ. The classic example is the torus and a coffee mug, both have genus 1.

2.6 Critical Phenomena in Cellular Automata

My study centers around finding *critical phenomena* caused by variations in topology of the kind described in Section 2.5. Critical phenomena denotes cases in which a subtle change in the definition of a cellular automaton causes a *phase transition* (e.g., a change in the Wolfram Class) in it's dynamics [16]. Interestingly, it seems that the idea of making variations in a cellular automatons definition this way receives very little attention in the literature, despite some studies indicating such changes can cause significant behavioural changes [16, 32, 33, 42, 55]. In terms of direct application, such results are useful in the study of system robustness [16].

Marr and Hütt performed a detailed quantitative study on the effect of changes in graph topology on 1-dimensional totalistic and outer-totalistic binary graph cellular automata rules [32, 33]. The types of topological perturbations they applied were Erdős-Rényi random graphs, delta-distributed random graphs, and scale-free graphs. Marr and Hütt did identify critical phenomena under these changes [32] and found that the number of complex rules (i.e., Wolfram Class IV) tends to decrease as the mean degree distribution increases [33]. Marr and Hütt proposed that their techniques could be used as probes for studying dynamical constraints imposed by topology in real biological metabolic networks [33].

Tomassini evaluated several graph topologies (i.e., small-world and scale-free) based on a cellular automaton's effectiveness in performing computational tasks (i.e., the density problem and the synchronisation problem) [55]. Tomassini concluded that by relating the classical topological regularity constraint, improved problem solving performance and robustness could be achieved.

Fatès studied the behaviour of cellular automata under random topological perturbations from a fault-tolerance perspective [16]. Fatès showed that topological perturbations can cause behavioural degradations. Minor topological changes

to Conway's game of life cause the "life-like" dynamics to completely degrade to a simple static regular pattern. Thus understanding the topological effect on dynamics is critical in the study of such a systems robustness.

The studies of Marr and Hütt [32, 33], Tomassini [55], and Fatès [16] are based on topological variations that can alter local neighbourhoods. Ventrella performed an interesting qualitative study on subtle dynamical differences induced by replacing a regular lattice with periodic boundaries with a geodesic grid [57], thus preserving local neighbourhoods and dynamics. For a rule similar to Conway's game of life, Ventrella observed phenomena in the geodesic grid that cannot occur in a regular lattice. Examples include perpendicular gliders crossing paths twice per orbit, and gliders changing direction without collision. Ventrella designed a novel XOR gate exploiting these phenomena and poses the question "*Are there unique spherical machines?*".

In my study, the quantitative techniques of Marr and Hütt [32, 33], Wolfram [62, 63] and Wuensche [74, 76] are applied to study topological effects of the type studied by Ventrella [57]. These topological changes are based on changing the genus of the 2-manifold (as discussed in Section 2.5) on which the cellular automaton evolves. The research questions presented in Section 1.3 have been constructed in order to gain insight into the question of the existed of machines that are unique to a specific genus of 2-manifolds.

2.7 Summary

This chapter has presented a review of the literature that forms a fundamental background to my study. A general overview of the theory of cellular automata has been presented, including the formal mathematical definition, computational properties, and the implications of these properties. The literature shows that by applying variations to the traditional definition of cellular automata, critical phenomena can be observed. To date, no quantitative study of such phenomena under topological variations which preserve local neighbourhood (and hence local dynamics) has been performed. My study fills this gap in the literature by investigating the statistical mechanics of cellular automata defined across topological spaces of differing genera.

Chapter 3

Methods

As presented in Section 2.6, quantitative studies shown that certain topological perturbations can cause critical phenomena to occur in graph cellular automata dynamics [16, 32, 33, 55]. However, all of these cases of critical phenomena have been observed using topological perturbations in which either homogeneity, size, and locality of neighbourhoods are not preserved. Ventrella's study [57] looked at the case in which homogeneity, size, and locality of neighbourhoods are preserved but the topology overall is *spherical* (genus 0) rather than *toroidal* (genus 1). As a result of qualitative observations, Ventrella proposes that there might exist machines unique to a spherical topology. This would suggest that a more intimate relationship exists between topology and dynamics than one can deduce from previously observed critical phenomena which can be largely explained due the changes in local dynamics.

A broad quantitative, experimental study was performed on the statistical mechanics of graph cellular automata under different topologies that largely preserve homogeneity, size, and locality of the cell's local neighbourhoods. Experiments have been designed to search out answers to the research question relating to the differences in the distribution of dynamical classes within rule space between topologies. This Chapter presents the details of the methods applied throughout this study.

3.1 Overview

The goal of any experimental study is to investigate if there is sufficient evidence to reject out hypothesis, that is the *null hypothesis* H_0. In this Section, H_0 will be constructed and the major steps in the research method are summarised. The details of each step will be presented in Sections 3.4, 3.5, 3.6.1,and 3.7.

Let A_n^N denote a graph cellular automaton with N cells and a graph G that is topologically equivalent to a connected 2-manifold of genus n, and let $R_{A_n^N}$ be the set of all possible rules. If the distribution of rule dynamical classes is denoted by $D\left(R_{A_n^N}\right)$, then it would seem appropriate to define the null hypothesis H_0 such that

$$H_0 : D\left(R_{A_0^N}\right) = D\left(R_{A_1^N}\right) = \cdots = D\left(R_{A_N^N}\right) : \tag{3.1}$$

Note that the last term in Equation (3.1) represents a fully connected graph (i.e., all cells are connected to all others). A fully connected graph forces the neighbourhood size to change, furthermore it has already been established that complex and chaotic totalistic rules cannot exist on a fully connected graph [32]. Since topologies that preserve local dynamics are of interest in this study, H_0 is further refined to be

$$H_0 : D\left(R_{A_0^N}\right) = D\left(R_{A_1^N}\right) = \cdots = D\left(R_{A_K^N}\right) \tag{3.2}$$

where $K << N$. Due to computational constraints $K = 2$ for the purposes of my project.

The overall experimental method consisted of sampling simulations of every rule in the rule space on genus 0, genus 1 , and genus 2 topologies (The techniques used in constructing the graphs of these topologies is given in Section 3.4). For every simulation entropy data was collected (See Section 2.3.3). Since there do not exist exact methods to classify a rule according to Wolfram's classification scheme [27, 28] a combination of the method applied by Wuensche [74] and the method applied by Marr and Hütt [32] has been used to approximate the dynamical distributions $D\left(R_{A_n^N}\right)$. Details of how these experiments we performed is given in Section 3.6.1.

In the literature, its seems rare for authors to justify their sample sizes when collecting entropy information [26, 28, 32, 32, 74, 76]. However, since it is requirement of this project that the entropy samples sufficiently approximate $D\left(R_{A,n}\right)$ it is appropriate that some testing is performed. A global entropy error analysis was performed for small cellular automata to try an extract an appropriate sample size as a percentage of size of the configuration space. Details of the method applied for this analysis is present in Section 3.5.

Since four difference entropy measures are collected per simulation, the distributions $D\left(R_{A,n}\right)$ are 4-dimensional. Very few non-parametric techniques for comparison of multivariate empirical continuous distributions exist [15, 36, 37]. The method that has been applied is based on the E-statistic originally formulated by Székely and Rizzo [51, 52], which is a non-parametric test based on the Euclidean distance between sample elements. The rationale behind selecting this technique and the details of its implementation are provided in Section 3.7.

3.2 Compute Resources

A detailed study of cellular automata dynamics requires more computational power than a standard desktop computing environment. This is largely due to the inherently parallel formulation of cellular automata being inefficient on a traditional Von Neumann based processor. Special purpose compute resources such as QUT's high performance computing facility are able to exploit more parallelism, thus enabling many thousands of simulations to be performed.

3.2.1 Hardware

Most of my experiments were executed on the QUT's compute cluster. This compute cluster is administered by the *high performance computing and research support group*[1] at QUT.

The current high performance computing platform[2] is a Silicon Graphics International Corp. (SGI)[3] Altix XE Computational Cluster with the following

[1] See http://www.itservices.qut.edu.au/researchteaching/hpc.
[2] Current at the time of writing.
[3] See http://www.sgi.com.

specification:

- 128 compute nodes.
- 1924 64-bit Intel Xeon Cores.
- 50:8 TeraFlop theoretical performance using double precision.
- 15; 264 GigaBytes of main memory.
- Infini-Band interconnect.
- SUSE Linux Operating System.

3.2.2 Software

Two software packages were heavily utilised in my project; they are GCALab and MATLAB®. GCALab[1] is an artifact of my project and is a custom appilcation for simluation and analysis of cellular automata. MATLAB® is a numerical software package and language developed by MathWorks, Inc.[2].

3.2.2.1 The Graph Cellular Automata Laboratory

The *Graph Cellular Automata LABoratory* (GCALab) is a special purpose, custom application designed for construction, simulation, visualisation and analysis of cellular automata defined on graphs (often graphically represented as surfaces). GCALab implements a interactive command line coupled with a parallel backend compute engine. A simple batch mode is also available for use with compute clusters.

GCALab started out it's life as several small independent C programs using a common cellular automata library that was developed in early stages of this project (called libGCA, see Appendix B). Eventually, small programs were reworked into a flexible and extensible framework system that could support rapid additions of new features and algorithms; the result was GCALab, a full description of it's design and functionality is provided in Appendix B.

[1] Available on GitHub https://github.com/davidwarne/GCALab.
[2] See http://www.mathworks.com.au.

GCALab has been heavily utilised as the cellular automata simulation engine, and analysis tool in my project. Computations such as exact global entropy calculations, Garden-of-Eden density, average attractor cycle length, and sampled Shannon entropy were all done using GCALab. GCALab combined with a shell scripting environment enabled scaling over 800 cores on QUT's HPC system.

3.2.2.2 MATLAB®

MATLAB® is a specialised software environment designed for numerical computation, data analysis, and visualisation. The MATLAB® interactive environment and scripting language provides a powerful and flexible tool for rapid algorithm prototyping, data analysis, and data exploration. MATLAB® is use widely in fields of science and engineering.

The main MATLAB® program/runtime implements and interpreter for the MATLAB® language. The MATLAB® language performs many linear algebra operations natively[1] and efficiently. Over the years many special purpose, domain specific toolboxes have been produced as "add-on" libraries implementing functions from image processing, optimisation, statistics, and many other fields.

In early stages of this project, MATLAB® was used as a prototyping environment to test calculations prior to implementing functionality into GCALab (see Section 3.2.2.1 and Appendix B). Whilst all of the data collection from cellular automata simulation was done using GCALab, some of the final analysis of the results was performed in MATLAB® due to is rich set of data analysis functions. Many of the plots and data visualisations shown in this thesis have also been generated in MATLAB® .

3.3 Graph Cellular Automata Construction

My project required a cellular automaton formalism that would allow for consistent definition across 2-manifolds of any genera. Since graph cellular automata (Section 2.1.2.3) are the most general definition, this is the formalism that was

[1] Hence the origin of the name MATLAB® from MATrix LABoratory.

implemented in the `libGCA` code library (See Appendix B). The graphs themselves are derived from triangular surface meshes generated using the techniques described in Section 3.4. Due to the method of defining G, this specific type of graph cellular automata is referred to as *triangular graph cellular automata*.

For a cellular automaton defined on a triangular tessellation, there are several possible neighbourhood definitions that could be applied to construct U. Two neighbourhoods definitions commonly used for rectangular tessellations are the *Von Neumann* neighbourhood and the *Moore* neighbourhood. The triangular tessellation analogs of these neighbour types are shown in Figure 3.1. The Von Neumann neighbourhood considers two cells to be neighbours if they share an edge, whereas the Moore neighbourhood considers cells that share a vertex to be neighbours.

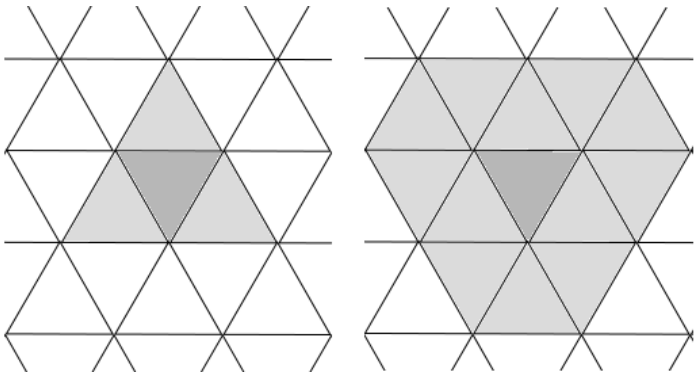

Figure 3.1: Neighbourhood types in the triangular tessellation. Left: Von Neumann neighbourhood. Right: Moore neighbourhood.

It seems that no such quantitative study looking at neighbourhood preserving topological changes exists to date. As a result, the main focus of my study has been the simplest possible 2-dimensional cellular automata; binary triangular graph cellular automata. More formally that is, graph cellular automata of the form $\mathsf{A}_{\mathsf{n}}^{\mathsf{N}} = \mathsf{f}\,\mathsf{G}_{\mathsf{n}}; \Sigma; \mathsf{U}_{\mathsf{VN}}; \Gamma\mathsf{g}$, where G_{n} is a graph topologically equivalent to a triangular mesh of genus n, $\Sigma = \mathsf{f}\,0; 1\mathsf{g}$, U_{VN} is the set of Von Neumann neighbourhoods, and Γ contains a single local state transition function $\mathsf{g} : \Sigma^4 \; ! \quad \Sigma$. This simplest case has the smallest possible rule space, thus allowing a more

in-depth study. However, for the sake of interest, a subset of the rule space has been investigated for binary cellular automata using the triangular Moore neighbourhood (i.e., "life"-rules [7]).

The local state transition function g is implemented using a look-up table (LUT). Each LUT lookup index l is constructed from the states of the cells in the neighbourhood. For the Von Neumann neighbourhood that is $l = (C(x_0); C(x_1); C(x_2); C(x_3))$ where x_0 is the centre cell and the x_1,x_2, and x_3 are the neighbours labelled arbitrarily. Thus there are $2^4 = 16$ possible neighbourhood configurations and $2^{16} = 65;536$ possible rules. For the Moore neighbourhood however, there are $2^{13} = 8192$ possible neighbourhood configurations and 2^{8192} possible rules, hence why only a small subset of this rule space has been explored [1].

Though the main research questions applied to the rule space as a whole, it is also of interest to know if certain subclasses of rules are affected more/less that then entire rule space. The specific subclasses of most interest are totalistic and outer-totalistic rules. This is largely due to the fact that relative orientation of adjacent local neighbours will not effect dynamics. For the Von Neumann neighbourhood, the Wolfram codes for totalistic rules are of the form

$$C_g = 32;768x_4 + 26;752x_3 + 5;736x_2 + 278x_1 + x_0 \tag{3.3}$$

where $x_i \ 2 \ f0;1g; i = 0;1;:::;4$, each constant is the sum of the neighbourhoods satisfying the respective total i set bits. Similarly the Wolfram code for outer-totalistic rules are of the form

$$\begin{aligned} C_g = {} & 32;768x_{3;1} + 16;384x_{3;0} + 10;368x_{2;1} \\ & + 5;184x_{2;0} + 552x_{1;1} + 276x_{1;0} + 2x_{0;1} + x_{0;0} \end{aligned} \tag{3.4}$$

where $x_{i;j} \ 2 \ f0;1g; i = 0;::3; j = 0;1$, each constant is the sum of the neighbourhoods with the centre cell of j and the outer total of i.

For the Moore neighbourhood, a special class of outer-totalistic rules were chosen for study. These rules are referred to as "life"-rules and are denoted by $E_l E_h F_l F_h$. This particular class was discovered by Carter Bays and are defined as

[1] Note that $2^{8192} \approx 10^{2466}$; there is an estimated 10^{80} hydrogen atoms in the universe. Good luck exhaustively simulating that rule space!

follows [7]: The pair $\mathsf{E_l E_h}$ is the *environment* rule which give the lower and upper bounds for the number of neighbouring live cells required for a currently live cell to remain alive; The pair $\mathsf{F_l F_h}$ is the *fertility* rule which give the lower and upper bounds for the number of neighbouring live cells required for a currently dead cell to come to life.

"Life"-rules are of interest since there are several known complex "life"-rules. Examples include rule 4644, 1868, and 3445 in the triangular tessellation [7] and the famous rule 2333 in the rectangular tessellation (i.e., Conway's game of life [60]). How these complex rules (and others of similar construction) behave under topological changes the preserve local neighbourhoods could provide insight into the interaction between dynamical complexity and structure.

3.4 Topology Generation

A fundamental component of this project was the construction of graphs which are topologically equivalent to 2-manifolds of a given genus. As discussed in Section 3.1, the scope of the project has been limited to genus 0, genus 1, and genus 2 topologies. This Section describes the method used to construct these graphs.

The method involves the construction of triangular surface meshes of the required topologies, the equivalent graphs are then extracted by setting mesh faces to graph nodes (the connectivity depends on the neighbourhood type used). The constraining factor in the mesh construction was the need to be able to build meshes of differing genera but with the same number of faces. This proved to be non-trivial, and in the end only genus 0 and genus 1 could be built with an identical number of faces; for genus 2 there is a difference of around 5%. For each topology, a base mesh with a minimum number of cells is constructed; larger meshes are then constructed via sub-division of the faces of the base mesh. For both the genus 0 and genus 1 topologies, the ideal base mesh was the Platonic solid the Icosahedron. The Icosahedron was selected partly due to it's use by Ventrella in construction of a geodesic dome (i.e., a genus 0 topology) [57], but it was mainly selected due to its property of being easy to remap to a torus. This

property is shown if Figure 3.2, the triangle strip that builds the icosahedron can be folded in orthogonal directions to build a flat torus.

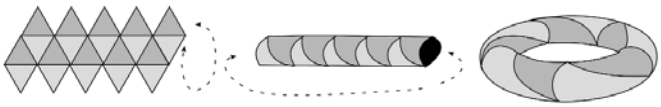

Figure 3.2: Remapping the triangular strip of the icosahedron to a *genus* 1 mesh.

The base mesh of the genus 2 topology (double torus) was a little more involved. The starting point was two genus 1 meshes, then two "sections" of each were removed leaving two "letter-C" looking objects. These two objects were then closed to form two tighter genus 1 meshes, which were then fused together to form the double torus. The method is shown in Figure 3.3.

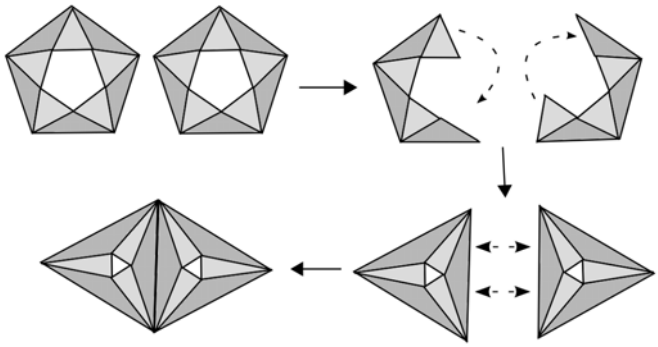

Figure 3.3: Construction of simple *genus* 2 mesh.

Once base meshes were constructed, larger meshes (and hence larger cellular automata) could be constructed via subdivision. A single subdivision consisted of taking a face and constructing a midpoint for each edge, denoted by m_i. These m_i can be connected to form a new triangle which divides the original face evenly into four new faces (See Figure 3.4).

The subdivision processes can then be applied iteratively to increase the mesh resolution (See Algorithm 1). This approach is both intuitive and elegant in

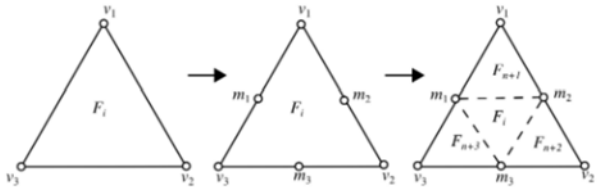

Figure 3.4: Mesh Subdivision.

terms of coding but it unfortunately has the side effect of producing duplicate vertices (consider two adjacent faces both applying the method). In the use case of this project, the overhead of merging duplicated vertices is worth the reduced development time of this method, as the mesh size is always relatively small thus mesh construction performance is dwarfed by the runtime of the simulations.

Algorithm 1 Mesh Subdivision

```
m ← CreateMesh(genus)
while : (m:numFaces = selectedResolution) do
  m ← Subdivide(m)
  m ← MergeDuplicates(m)
end while
```

The topology mesh generation technique described in this Section has been implemented as a part of the `libMesh` C library, which is utilised by the `libGCA` C library (See Appendix B). Examples of typical output is shown in Figure 3.5; note that geometrically these meshes do not look "smooth" but this not a problem as the actual geometry is essentially discarded[1].

[1] Except when `GCALab` is run in graphics mode.

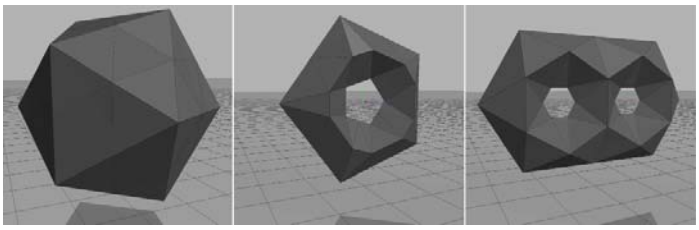

Figure 3.5: Outputs of `libMesh` topology generation function.

3.5 Error Analysis of Entropy Measures

In the literature, justifications for sample sizes for entropy calculations are rarely provided [28, 32, 33, 74]. In order to have confidence that the sampled entropy values sufficiently approximate $D(R_{A,n})$ some analysis on the appropriate sample size was necessary. This analysis involved computation of the exact *global entropy* and comparing approximation errors as a function of sample size (i.e., the relative sample size compared to the entire configuration space).

3.5.1 Computing the Exact Global Entropy

Let the *global entropy* of cell x_i be defined to be the Shannon entropy given in Equation (2.4) with an subtly different method of computing the probabilities of the "micro-states". In the case of global entropy, the p_j^i measure the probability that cell x_i is in state j over all possible *complete transient evolution* of $\mathbf{A}$. Let a complete transient evolution be defined to be one with a Garden-of-Eden initial configuration and terminating at the start of an attractor cycle (i.e., an evolution with traverses a path from leaf to cycle in the configuration transition graph), as shown in Figure 3.6. This particular entropy measure was defined in order to maximise the error caused by small sample sizes.

By traversing every complete transient evolution of $\mathbf{A}$ the exact frequency of every configuration can be accumulated without unnecessary repetition. To compute the global entropy of $\mathbf{A}$ exactly, every Garden-of-Eden configuration is required to be known. An efficient algorithm (i.e., best case in $\Omega(n)$, worst case in $O(n^3)$, and average case in $\Theta(n^2)$) for the detection of Garden-of-Eden

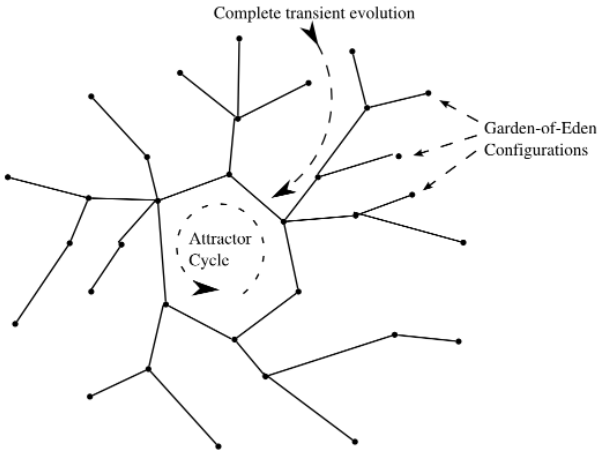

Figure 3.6: A *complete transient evolution*.

configurations was developed in order to perform the global entropy calculation (See Appendix A). Probabilities were calculated exactly by counting the state frequencies of each cell over every possible complete transient evolution of $\mathbf{A}$.

Even with an efficient method of determining Garden-of-Eden configurations, the total global entropy of $\mathbf{A}$ is computationally intractable for large cell counts as every configuration need to be tested (i.e., complexity class $\Theta(2^n)$). Through the utilisation of QUT's HPC cluster, it was possible to compute the exact global entropy of every elementary cellular automaton for cell counts $n \in \{8; 16; 32\}$.

3.5.2 Computing Sample Global Entropy

Having computed the exact global entropies for small elementary cellular automata, the next step was to identify an approximation error for different sample sizes. Approximations were computed by a random selection of N initial conditions (i.e., completely ignoring the Garden-of-Eden test) and evolving the cellular automaton to the start of an attractor cycle. Cell state frequencies were collected over all N samples to approximate the exact probabilities as calculated in Section 3.5.1.

Approximations were computed for every elementary cellular automaton for samples sizes ranging from □ 0:000006% to □ 25% of the total configuration space size. Results from these experiments provided a guide toward selection of sample sizes and cellular automaton cell counts used for the approximation of D (R$_{\mathbb{A}}$) computed according to the methods described in Section 3.6. The details of the error analysis results are presented in Section 4.1.

3.6 Computation of Dynamical Distributions

Approximations for the dynamical distributions D R$_{\mathbb{A}}$ $_0^N$,D R$_{\mathbb{A}}$ $_1^N$, and D R$_{\mathbb{A}}$ $_2^N$ were computed by repeated simulation of cellular automata sampled from the rule spaces R$_{\mathbb{A}}$ $_0^N$,R$_{\mathbb{A}}$ $_1^N$, and R$_{\mathbb{A}}$ $_2^N$ respectively. These approximate dynamical distributions essentially represent a global view of all cellular automata dynamical possibilities for each topological genus, which are compared using statistical methods (See Section 3.7). In this Section, the details of the method of simulation, the types simulations, and the derivation of the dynamical distributions is presented.

3.6.1 Classification of Rule Space

Combinations of entropy metrics (Discussed in Section 2.3.3) provide an excellent method for quantifying a cellular automaton's dynamical class [28, 32, 62, 63, 74]. Though exact Wolfram class[1] classification is not directly computable, a rough classification of the rule space is possible. Three particularly useful entropy metrics for the purposes of automatic rule space classification are the so-called *Shannon entropy*, *Word entropy*, and *Input entropy* given in Equations (2.4),(2.7), and (2.8) (All from Section 2.3.3) respectively.

The Shannon and Word entropies (averaged over all cells) were applied by Marr and Hütt for the classification of rule space in their study on the effect of random graph topology changes have on the rule space of binary graph cellular automata [32, 33]. By plotting the average Shannon and Word entropies against each other, a visualisation of the rule base can be produced. Marr and Hütt

[1] If there is such a thing.

identified distinct regions within this visualisation roughly match the Wolfram classes.

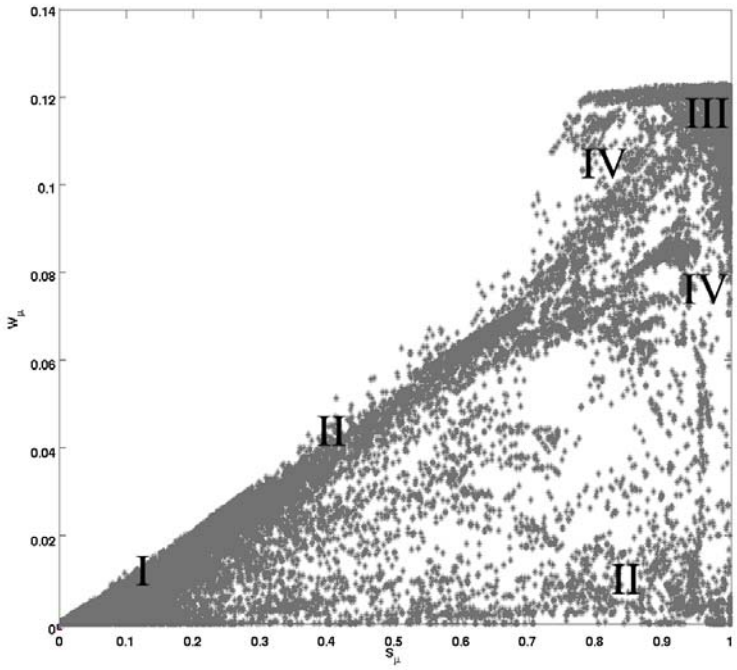

Figure 3.7: Rough classification of rule space using the average Shannon $\mathsf{S}_{\square}$ and Word $\mathsf{W}_{\square}$ entropies.

These rough regions are shown in Figure 3.7. It should be noted that exact boundaries between these regions do not exist, instead there is a smooth transition between regions. Furthermore these regions really represent the most common dynamical class in that region (e.g., A Class IV cellular automaton may exist in the region labelled Class III).

Wuensche applied a similar rule space classification technique, but he only applied the Input entropy [74]. Wuensche visualises the rule space by plotting the average Input entropy against the varience of the Input entropy. Regions analogous to the Wolfram classes using the Input entropy are shown in Figure 3.8.

The Input entropy method is extremely good at locating cellular automata with complex dynamics (i.e., Class IV).

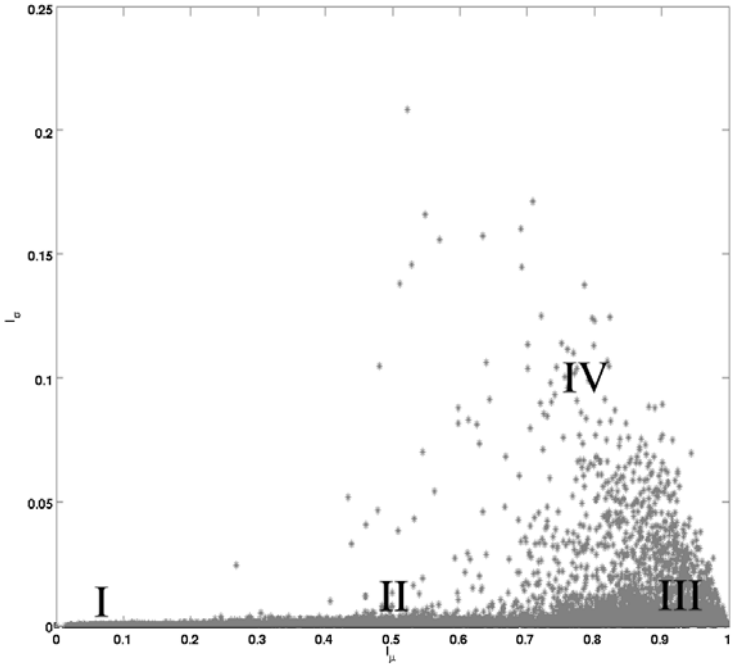

Figure 3.8: Rough classification of rule space using the mean $\mathsf{I}_{\square}$ and variance $\mathsf{I}_{\square}$ of the Input entropy.

Since both these techniques provide a reasonable method to quantify the dynamical distribution of a given rule space, it seemed logical to take a combination of the two. For every cellular automaton simulation the average Shannon $\mathsf{S}_{\square}$, Word $\mathsf{W}_{\square}$, and Input $\mathsf{I}_{\square}$ entropies and the variance of the Input $\mathsf{I}_{\square}$ entropy were computed. This results in a 4-dimensional summary of the dynamical characterisation of every rule in the rule-space. Thoughout this thesis such a summary is referred to as the dynamical distribution of a rule-space, as it roughly indicated the frequencies of the Wolfram classes.

3.6.2 Other Metrics

Look-up table analysis and configuration graph analysis (as discussed in Sections 2.3.1 and 2.3.2) also provide dynamical information comparable across topologies. For each entropy simulation run, look-up table parameters and configuration graph properties were computed. These quantities were collected to complement the findings of the statistical analysis of dynamical distribution comparison, and to reinforce findings. Rigorous statistical tests were not applied to these quantities.

The look-up table parameters applied were Langton's □-parameter [28] and Wuensche's Z-parameter [74]. The □-parameter is trivial to compute, since it utilises the cellular automaton's look-up table only. Furthermore, since the look-up table does not change across topologies which preserve local neighbourhoods, then the □-parameter of a particular rule will remain constant. The Z-parameter, however, does take the automaton's topology into account, so minor changes do occur. The usage of □ and Z are typically used as a rough classification method, so finding critical phenomena across topologies may have implications for their applicability to all topologies.

Configuration transition graph properties are extremely computationally expensive to compute exactly. As a result approximations were based on the same samples as that of the entropy computations. Properties such as the G-density [76], mean cycle length, and mean transient [26, 68, 74] were sampled. These properties were collected to compare with results of the entropy comparisons in the hope that it may provide more insight (despite the limited accuracy of these properties).

3.6.3 Sampling Method

Not every single cellular automaton in a given rule space need be sampled to get an exact picture of the dynamical distribution. There are many equivalence classes within a rule space which can be exploited. By exploiting equivalences the number of rules is effectively halved.

The most significant equivalence is that of "bit-flip" equivalence. An automaton with a quiescent state $□_q = 0$ will be dynamically identical to another

automaton with $\square_q = 1$ in which the rule code is a "bit-flip" and reverse order of the first. For example rule 110 is equivalent to rule 137 in this respect. By only simulating automata with a $\square_q = 0$, this can be done by selecting only rules with even Wolfram codes.

In order to get as accurate a representation of dynamical distributions as possible, as much of rule space as was computationally feasible was sampled. For the Von Neumann neighbourhood case effectively every rule could be represented (as there is only 2^{16} of them), however, only a very small portion rule space induce by the Moore neighbourhood could be sampled. In this case only a special case of outer-totalistic rules, so-called "life" rules [7], were sampled (albeit completely, as there is only 8484 of them).

For each sampled cellular automaton around $100;000$ entropy samples were taken based on the results of experiments to determine error in approximate entropies (See Section 4.1.2). Each entropy sample consists of a random initial condition followed by a simulation. Entropy measures from each simulation run were averaged to get a single 4-tuple representing the dynamics of that rule. Every simulation was repeated for genus 0,1, and 2 topologies.

To ensure stable entropy measurements, every simulation executes a "lead-in" period (i.e., a number of initial time-steps which do not contribute to the entropy calculations). This period was equal to half the total number of time-steps that contribute to the entropy calculations. For example, to compute entropies over 2000 time-steps, the simulation will execute 3000 time-steps, discarding the first 1000.

3.6.4 Simulation Cases

Several simulation cases, that is a rule-type/neighbourhood-type pair, were investigated. In each simulation case, samples as described in Section 3.6.3 were take to compute a dynamical distribution for that case. Some cases are subsets (and hence so are there dynamical distributions) of each other whereas others are distinct.

The for the Von Neumann neighbourhood the entire rule space could be covered, this represents one large simulation case (named VN_ALL). Here a complete

Table 3.1: Simulation Cases.

Simulation case	Cells	Sampled rules	Samples/Rule	Time-steps/Sample
VN_ALL_80	80	32; 768	100; 000	3000
VN_TOT_80	80	32	100; 000	3000
VN_OUTTOT_80	80	256	100; 000	3000
VN_ALL_1280	1280	32; 768	1; 000	5000
VN_TOT_1280	1280	32	1; 000	5000
VN_OUTTOT_1280	1280	256	1; 000	5000
M_LIFE_1280	1280	8484	1; 000	5000

global view of the dynamical distributions are calculated. The VN_ALL simulation case is of the most use for testing the most general form of the *null* hypothesis in Equation (3.2).

Von Neumann neighbourhood simulation cases which are subsets of the VN_ALL case are the totalistic rules (named VN_TOT) and the outer-totalistic rules (named VN_OUTTOT). Both these simulation cases are of interest as these rule do not suffer from orientation problems (which could have an effect on results), and there applicability to applications is wider. Comparing the effects of topology on these classes separately proves to yield some interesting (and rather surprising results, see 4.2).

For the Moore neighbourhood only the one simulation case was taken, that of "life" rules [7] (named M_LIFE). This case was taken specifically to compare with the outer-totalistic case on the Von Neumann neighbourhood, VN_OUTTOT. By making this comparison, hints as to how the topological effect changes with neighbour type can at least be speculated.

A summary of all the simulation cases, and their respective cell numbers and sample numbers are given in Table 3.1. Note that the sample sizes are small for larger cell counts $\mathsf{n} = 1280$. This is in part due to computational constraints and due to the fact that trillions of samples would be require to even get close the percentage of configuration space covered for $\mathsf{n} = 80$.

3.7 Comparison of Dynamical Distributions

Comparing empirical data distributions against a known statistical distribution is a routine task in inferential statistics. However, comparing two empirical distributions for equality (called the K-sample test) without assuming any underlying distribution is more complex. Tests that do not require a known statistical distribution are called *distribution-free*. To test for evidence against the *null* hypothesis (Equation (3.2)), a multivariate distribution-free K-sample test is required.

3.7.1 Equality tests for Empirical Distributions

Univariate cases of the K-sample test are traditionally solved using distribution-free methods such as the Kolmogorov-Smirnov test [17, 46]. Unfortunately, the Kolmogorov-Smirnov test does not generalise directly to the multivariate case (due to the non-uniqueness of joint cumulative probability density function definitions). Some studies have attempted to find algorithmic approximations to a multivariate Kolmogorov-Smirnov test [15, 36].

In more recent years, distribution-free tests have been developed based on the Euclidean distances between data points. Examples of such tests are Rosenbaum's cross-match test [37] and Székely and Rizzo's E-statistic [51, 52]. For the purposes of this project, the method of Székeley and Rizzo was selected due to the existence of limit theorems pertaining to the error in the method [51].

3.7.2 The E-statistic

The E-statistic is a test statistic for testing the equality of K multivariate sample distributions [51]. The E-statistic is based on the Euclidean distance between sample points.

Let X and Y be random variables in R^d of unknown distributions D_X and D_Y. Given two random samples $x_1; ...; x_{n_1}$, and $y_1; ...; y_{n_2}$ of X and Y then the

two-sample E-statistic e(X;Y) is given by

$$E_n = e(X;Y) = \frac{n_1 n_2}{n_1 + n_2} \left(\frac{2}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \|x_i - y_j\| - \frac{1}{n_1^2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} \|x_i - x_j\| - \frac{1}{n_2^2} \sum_{i=1}^{n_2} \sum_{j=1}^{n_2} \|y_i - y_j\| \right) \tag{3.5}$$

where $n = n_1 + n_2$. Note that $E_n = 0$ if the two samples are identical.

For the K-sample test consider the random samples $X_1, \ldots, X_K$ with $n_i = |X_i|$. The two-sample statistic, given in Equation (3.5), can be extended to the K-sample test can be defined by summation of two-sample tests for each of the $\frac{K(K-1)}{2}$ possible sample combinations. The resulting form for the K-sample E-statistic is

$$E_n = \sum_{1 \leq i < j \leq K} e(X_i; X_j) \tag{3.6}$$

where $n = n_1 + \ldots + n_K$.

Let $\Pr(E_n > e)$ denote the limiting probability of E_n for $n \to \infty$. It was proven by Szekely and Rizzo that given $\alpha \in (0,1)$ then there exists a c_α such that $\Pr(E_n > c_\alpha) = \alpha$ [51]. Thus a statistical test can be defined that rejects the *null* hypothesis $H_0 : D_{X_1} = D_{X_K}$ at some statistical significance level α if $E_n > c_\alpha$. As shown in Section 3.7.3, the exact form of the distribution $\Pr(E_n > e)$ is not required, hence the technique is distribution-free.

3.7.3 The E-Test for Equality of Empirical Distributions

In practice, the exact form of $\Pr(E_n > e)$ is not known. An approximation which is accurate up to a given significance level can be computed using a permutation test [13]. The permutation test involves repeated random relabelling of the sample points from the K samples $X_1, \ldots X_K$ which are pooled into the one list. The accuracy and computation time of the test is dependent on the number of *bootstrap* samples taken (i.e., the number of re-samples).

The permutation test proposed by Székely and Rizzo [51] takes the random samples $X_1, \ldots, X_K$ from the distributions $D_{X_1}, \ldots, D_{X_K}$ respectively. These samples are combined to form a pooled sample $\{Y_1, \ldots, Y_n\} = X_1 \cup \ldots \cup X_K$. The

number of re-samples B is selected such that $(B + 1)$ □ 2 Z. The E-statistic E_n^b is computed for every bootstrap sample $f Y_1^b; :::; Y_K^b g$, $b = 1; :::; B$, using the new K-samples $X_1^b; :::; X_K^b$ where

$$X_i^b = f Y_{m_i □ 1 + 1}^b; :::; Y_{m_i}^b g; i = 1; :::; K \tag{3.7}$$

$$m_j = \sum_{i=1}^{j} n_i \tag{3.8}$$

$$m_0 = 0: \tag{3.9}$$

After evaluation of E_n^b for $b = 1; :::; B$, the following bootstrap estimate can be applied:

$$\Pr(E_n > e) □ \frac{1}{B} \sum_{b=1}^{B} I(E_n^b > e) \tag{3.10}$$

where $I(E_n^b > e)$ evaluates to 1 if $E_n^b > e$. Finally the test statistic E_n^0 from the original random samples $f X_1; :::; X_K g$ is computed, which enables the computation of the p-value,

$$p = \frac{1}{B} \sum_{b=1}^{B} I(E_n^b > E_n^0): \tag{3.11}$$

In general, if $p <$ □ then it can be said that sufficient evidence exists to reject $H_0 : D_{X_1} =; :::; = D_{X_K}$.

3.7.4 Application of the E-Test to Dynamical Distributions

As described in Section 3.6, random samples we collected of cellular automata evolutions defined on genera 0; 1; and 2 topologies. This resulted in a set of samples $X_0; X_1; X_2$ in which X_i □ R^4. In all simulation cases (i.e., a combination of rule type and neighbourhood type; See Section 3.6.4), the sample sizes were all equal; that is, $n_0 = n_1 = n_2$. The pooled sample size n ranged from 48 to 98; 304 depending on the simulation case.

Due to the sample sizes involved for some simulation cases (such as the pooled sample size of $n = 98; 304$ generated by all binary rules types with Von Neumann

neighbourhoods), the code implementation provided by Székely and Rizzo [1] was not suitable. As a result, a MATLAB® code port of the K-sample E-test was developed to balance memory and compute time. The optimised MATLAB® version provided the capability to run test without the need to use QUT's HPC resources. The MATLAB® code used for the E-test is given in Appendix C.

For each simulation case, the E-test was applied to the 3-sample test with the samples $X_0; X_1; X_2$ drawn from the Dynamical Distributions $D(R_{A,N_0})$, $D(R_{A,N_1})$, $D(R_{A,N_2})$. The *null* hypothesis being tested asserts that the dynamical distributions are equal (Equation (3.2) with $K = 2$). In each test, B was selected of sufficient size to support a significance level of 5% (that is $\alpha = 0.05$). The results of these tests a given in Section 4.2.

3.8 Summary

In this chapter, the methods of analysis, and resources required for this study have been presented. From formulation of the *null* hypothesis to the simulation of cellular automata to the statistical analysis of dynamical distributions, each component plays in study of topology and is effect on cellular automata dynamics. The fine details of each component of the study has been given alongside the context of that component within the study as a whole.

A preliminary error analysis of the global entropy measure was performed to inform the choice of sample sizes in dynamical distribution sampling. The development of an efficient method for detecting Garden-of Eden configurations greatly reduced overall runtime. The results of this analysis are presented in Section 4.1.

To simplify the study only binary cellular automata on the triangular tessellation were studied, though the formalism used was that of graph cellular automata to support topological variation. Approximations of dynamical distributions of rule spaces induced by the triangular Von Neumann and Moore neighbourhoods were computed using entropy metrics from many 1,000's of simulations. Dynam-

[1] Downloadable from http://cran.r-project.org/web/packages/energy/index.html.

ical distributions for topologies of genus 0, genus 1, and genus 2 were computed in all cases.

The *null* hypothesis was constructed to assert the equality of the dynamical distributions of cellular automat rule spaces regardless of topological genus. Dynamical distribution equality was tested using a distribution-free statistical test based on the Euclidean distance called the E-test. The results of this test are given in Section 4.2.

These methods have been presented to communicate the work that was performed throughout the course of this project. Sufficient information on all methods has been given for an interest reader to repeat and test the results of this work. The results of these methods and what they reveal about the research questions are present in Chapter 4.

Chapter 4

Results and Discussion

In this chapter, the results of all experiments performed on this project are presented. The initial entropy error analysis performed to infer required sample sizes for entropy calculations are given in Section 4.1. The resulting dynamical distributions derived from many sampled entropy calculations are compared across different topologies from a qualitative and quantitative perspective in Section 4.2. Some interesting examples of critical phenomena are given in Section 4.3. Finally the interpretations of results are discussed in Section 4.4.

4.1 Entropy Error Analysis

The reasoning behind selection of a certain sample size for entropy calculations seem to be rarely presented in the literature [28, 32, 33, 74]. Perhaps, this is often seen as "obvious" or common knowledge. It was decided that for this project an estimation of expected errors in the entropy calculations should be computed to inform the choice of sample size. The methods applied to compute the global entropy error for a given sample size is provided in Section 3.5. In this section, the results are presented and a discussion of how this informed the sample size and the impact on expected error bounds follows.

4.1.1 Computed Errors for Elementary Cellular Automata

The exact global entropy was computed for every elementary cellular automaton for cell counts $n = 8; 16; 32$. Entropy errors were computed for each automaton using a large range of sample sizes which were chosen to represent a large range in percentage of configuration space. The purpose of the selection was to test if accuracy is dependent of sample size alone or whether it is also affected by the size of the configuration space $j\Phi j = j\Sigma j^n$.

Sample size ranges varied for different cell counts. Automata with cell counts $n = 8; 16$ have a relatively small configuration space. Hence, it was trivial to compute sample sizes up to 25% (i.e., $N = 0{:}25j\Phi j$)of the configuration space. However, it was not computationally feasible to take sample sizes of this size for $n = 32$, thus only samples size of up to 0:25% of configuration space were taken.

For each cell count and sample size the resulting global entropy errors where averaged to compute an expected absolute error for that case. This expected error is plotted against sample size as a fraction of configuration space in Figure 4.1. It is clear from Figure 4.1 that whilst each cell count case shows the same downward trend as the fraction of configuration space sampled increases, there seems to be no correlation between the different cases for a given fraction. It may be reasonable to conclude from this that configuration space size is not the dominant factor in trend of the errors.

This conclusion is further supported when the expected errors are plotted against the actual sample size rather than the fraction of configuration space. In Figure 4.2, the trends between the different cell counts seem much more correlated. In all cases, the expected error seems to converge to a steady state at around $N = 10^3$. Interestingly, the error for $n = 32$ seems to converge much sooner than $n = 16$; The reason for this is unknown, but it may have been due to a particularly bad set of random samples. Ultimately, this can only be confirmed by many repeated trials.

Though it seems that raw sample size seems to be the main driver of the trend in expected error. There is still another notable difference between the cases. The "steady state" reached in the error seems to increase as the configuration size increases. The minimum reached by $n = 8; 16$ are about equal at $E = 0{:}025$,

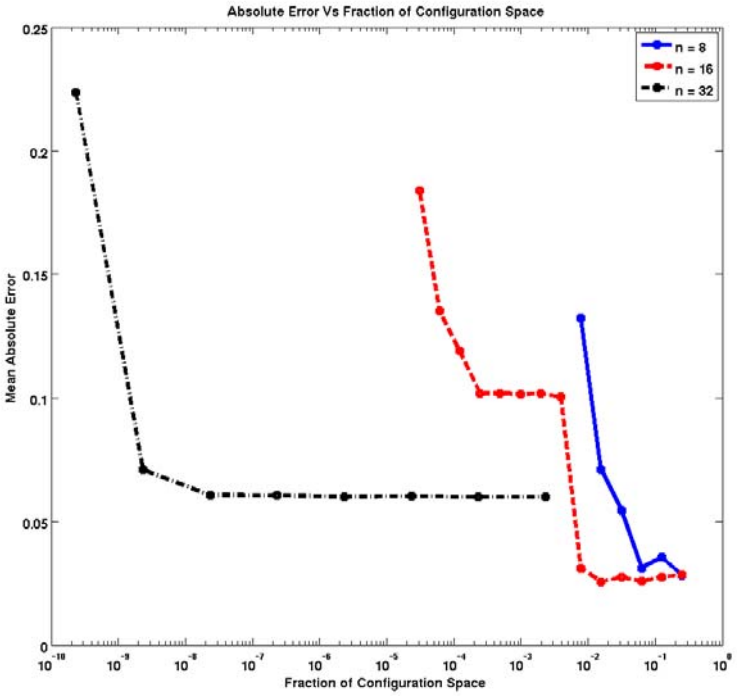

Figure 4.1: The global entropy error compared with sample size as a fraction of configuration space.

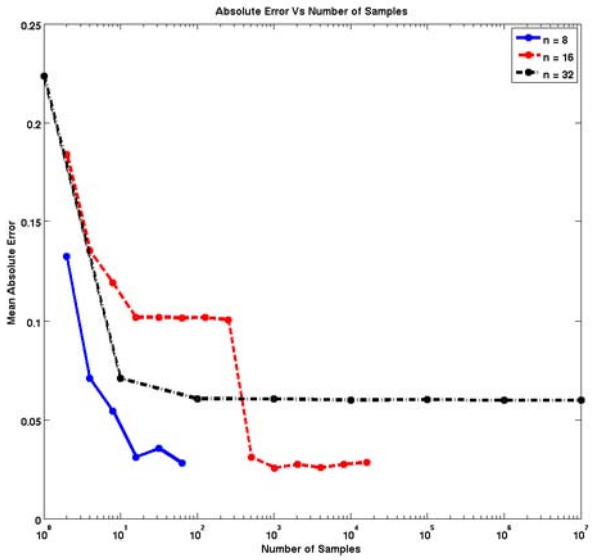

Figure 4.2: The global entropy error compared with raw sample size.

however for $n = 32$ the error never seems to get below $E = 0{:}065$. Of course, this barrier would eventually be broken as $N \gg |\Phi|$.

The ultimate conclusion from this analysis was that the expected entropy error is affected by sample size and configuration space size. The factor with the greatest effect on the error is clearly the sample size, but the size of configuration space must still be considered.

4.1.2 Implications for Sample Size Selection

The entropy error analysis reveals that the sample size selected for entropy approximations cannot be selected in complete isolation from the size of the configuration space. When selecting the sample size in practice, one must also consider the available compute resources available.

As discussed in Section 3.6.4, several different simulation cases were selected. Each of these cases uses cellular automata with either 80 cells or 1280 cells. From Figure 4.2 in Section 4.1.1, it may have seemed reasonable to simply set

$N = 1;000$ in all cases. However, since the simulation cases with 80 cells require less computation to simulate, it is reasonable to increase the sample size to ensure the error reaches the "steady state'. As a result sample sizes of $N = 100;000$ were selected for approximation of entropy measures.

The increase in computation required to simulate Automata with 1280 cells meant that it was not practical to increase the sample size much beyond $N = 1;000$. Since the configuration space for this case is so gigantic, it is likely that larger errors can be expected for this case.

Both simulation cases of $n = 80$ and $n = 1280$ have a significantly larger configuration space than the cases selected for entropy analysis. Unfortunately this could not be avoided as $n = 80$ is too large to compute the exacted global entropy and $n = 32$ is too small to construct a topology of genus 1 or 2. Since the error seems to increase from $n = 16$ to $n = 32$, it is expected that there will be an increase in error.

Without a much more rigorous analysis, how this error is expected to increase as the configuration space size increases is unknown. However, it can be seen from Figure 4.2 that the error for a very small sample size $N = 2$ seems to be converging. This error of $E = 18\%$ could be considered as the absolute maximum expected error. Since the error certainly decreases quickly as N increase at this point, it would be reasonable to conjecture that the overall expected error for the interval $N \ 2 \ [10^3; 10^9]$ is in the range $E \ 2 \ [0{:}065; 0{:}1]$. For the sake of this project an error of around $E = 0{:}08$ is considered to be expected.

4.2 Comparison on Rule Space Classifications

4.2.1 Visualisation of Dynamical Distributions

Since the dynamical characteristics each cellular automaton in a rule space is represented by a 4-tuple $(S_□; W_□; I_□; I_□)$, the dynamical distribution is 4-dimensional. Visualisation of these 4-dimensional structures for qualitative comparison is non-trivial. Rather than try to visualise the complete shape of these distributions, the dynamical distributions will be visualised in the two planes defined by the axis pairs $(S_□; W_□)$ and $(I_□; I_□)$. Due to the prior work in these planes [32, 33, 74, 76]

(See Section 3.6.1), it is possible to interpret differences between dynamical distributions in terms of Wolfram's classification scheme [63].

4.2.1.1 Entire Rule Space

For the simulation cases involving the Von Neumann neighbourhood (See Section 3.3) it was computationally feasible to compute dynamical characteristics of every cellular automaton in the rule space. There are 2^{16} rules in each of these rule spaces. The simulation results of these cases represent the dynamical distributions $D\left[R_{A,i}^{80}\right]$ and $D\left[R_{A,i}^{1280}\right]$ respectively with $i \in [0;1;2]$. These results are shown in Figures 4.3 and 4.4.

Qualitatively, there are certainly differences in dynamical distributions across genera for $n = 80$ (See Figure 4.3). The most visually evident of these are in $S_\lambda W_\lambda$ plane in the regions $0{:}4 < S_\lambda < 0{:}7; 0 < W_\lambda < 0{:}015$ (Class I/Class II), $0{:}9 < S_\lambda < 1{:}0; 0 < W_\lambda < 0{:}02$ (Class II), $0{:}5 < S_\lambda < 0{:}8; 0{:}1 < W_\lambda < 0{:}12$ (Class IV/Class III). Though not quite as obvious, in the $I_\lambda; I_\lambda$ plane as contains a visual difference around $0{:}7 < I_\lambda 0{:}9; 0{:}1 < I_\lambda < 0{:}15$ (Class IV/Class III).

Interestingly the two most visually identifiable differences in both planes are both located on a rough boundary region of Class IV and Class III cellular automata. In both cases the genus-1 topology seems to have a much higher density of rules located in these regions. Using Figure 4.3 alone, it would seem that a rule space $R_{A,1}^{N}$ yields a higher population of rules located near the "edge-of-chaos", that there is a general trend that Class II cellular automata have much lower Word entropies (i.e., less variation in oscillatory periods).

Unfortunately, the trends evident in the case for $n = 80$ do not seem to as n increases. This can been seen in Figure 4.4, in which $n = 1280$. For this case there is no clear visual difference between the dynamical distributions across topological genera. A more detailed discussion of causal factors is given Section 4.4.

4.2.1.2 Totalistic Rules

Totalistic rules were only simulated for the Von Neumann neighbourhood via the simulation cases VN_TOT_80 and VN_TOT_1280. In these cases only 2^5 possible

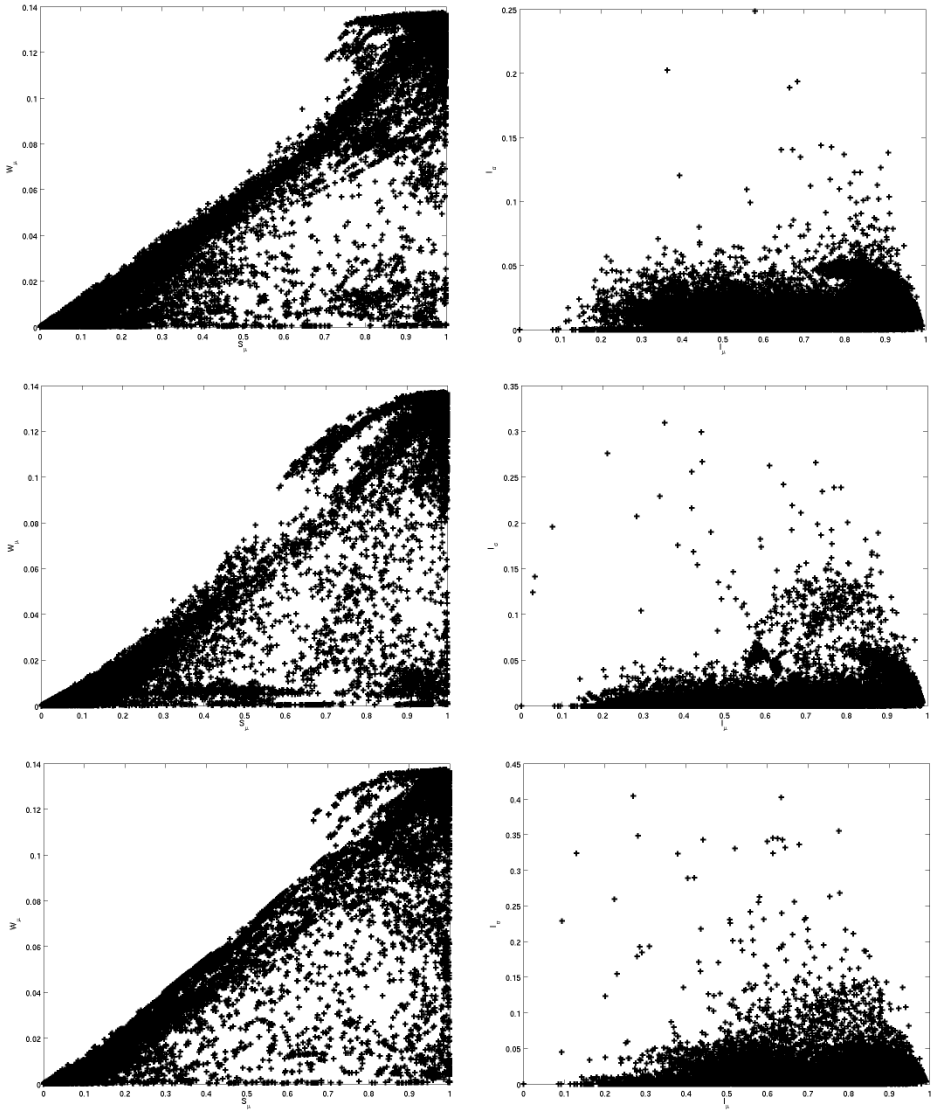

Figure 4.3: Dynamical Distributions of $\mathsf{R}_{\mathrm{A}\,0}^{80}$ (top),$\mathsf{R}_{\mathrm{A}\,1}^{80}$ (middle), and $\mathsf{R}_{\mathrm{A}\,2}^{80}$ (bottom) using a Von Neumann neighbourhood.

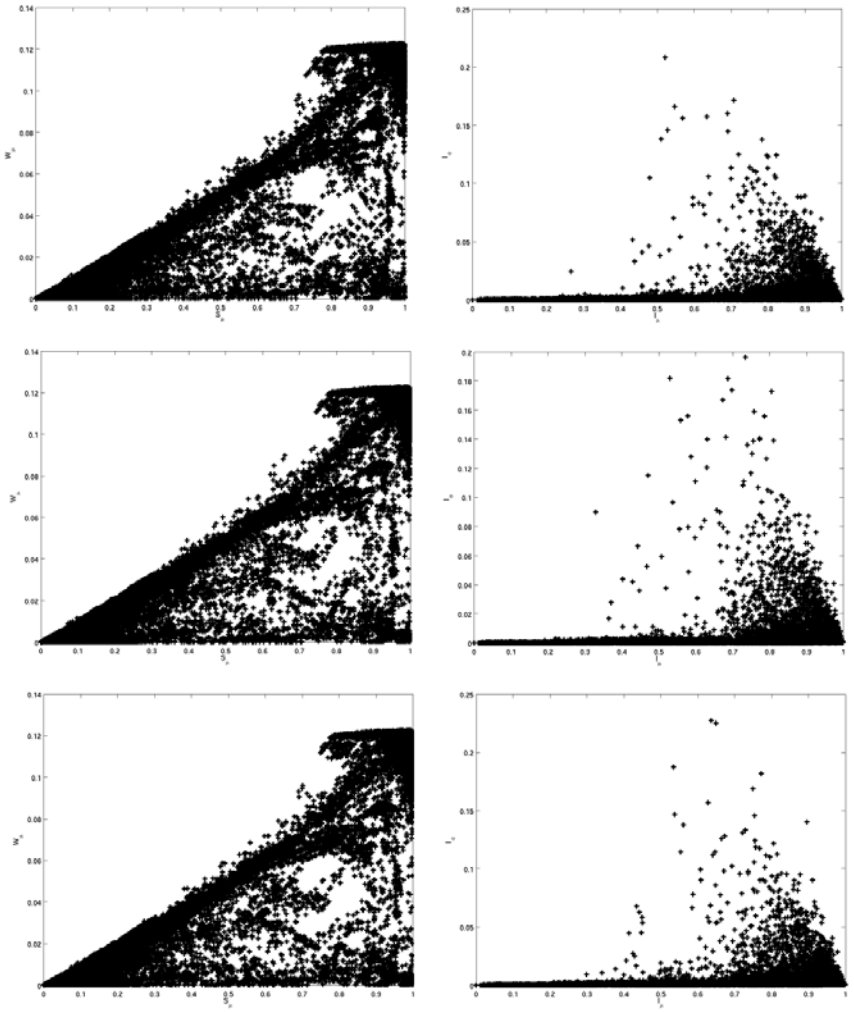

Figure 4.4: Dynamical Distributions of $\mathsf{R}_{\mathbb{A}\,0}^{1280}$ (top),$\mathsf{R}_{\mathbb{A}\,1}^{1280}$ (middle), and $\mathsf{R}_{\mathbb{A}\,2}^{1280}$ (botton) using a Von Neumann neighbourhood.

rules exist. The dynamical distributions for these totalistic rules are shown in Figures 4.5 and 4.6.

In both simulation cases, there does not seem to be any visual difference between totalistic rule spaces across different topological genera. However, it is still true that there is visually less variation in the case for $n = 1280$ (Figure 4.6) over $n = 80$ (Figure 4.5). This is consistent with the behavior observed in Section 4.2.1.1.

4.2.1.3 Outer-totalistic Rules

For the outer-totalistic simulation cases for both Von Neumann and the Moore neighbourhood types were investigated. There are 2^8 possible rules in the Von Neumann rule spaces, which is computationally feasible to explore entirely. However, there are 2^{24} possible outer-totalistic rules for rule spaces in the Moore neighbourhood so only a subset of these (the so-called "life" rules [7], of which there are $10;000$) were sampled. The results of these simulations are shown in Figures 4.7 and 4.8 for the Von Neumann neighbourhood and Figure 4.9 for the Moore neighbourhood.

Dynamical distributions are visually different between topological genera for the case of $n = 80$ (Figure 4.7). This is most evident differences in the $S_{\square}; W_{\square}$ plane are in the regions $0 < S_{\square} < 0{:}3; 0 < W_{\square} < 0{:}03$ (Class I/Class II/) and $0{:}6 < S_{\square} < 1{:}0; 0{:}06 < W_{\square} < 0{:}12$ (Class II/Class IV/Class III). In the first region, rule spaces defined on a genus-0 or genus-1 topology seem to yield a higher population of rules. The second region is dominated by rules defined on a genus-0 or genus-2 topology; this seems to indicate that there are fewer rules around "edge-of-chaos". This is also reinforced by the fact that there are very few genus-1 rules with $I_{\square} > 0{:}05$ in the $I_{\square}; I_{\square}$ plane.

Just as in Sections 4.2.1.1 (i.e., the entire rule space) and 4.2.1.2 (i.e.,totalistic rules), there is no visual (i.e., qualitative) difference in the dynamical distributions across rule spaces defined on different topological genera for larger automata with $n = 1280$ Figure 4.8. However, this becomes more interesting when we consider the comparison of dynamical distributions for outer-totalistic rule spaces using a Moore neighbourhood. The comparison distribution plot for the so-called "life"

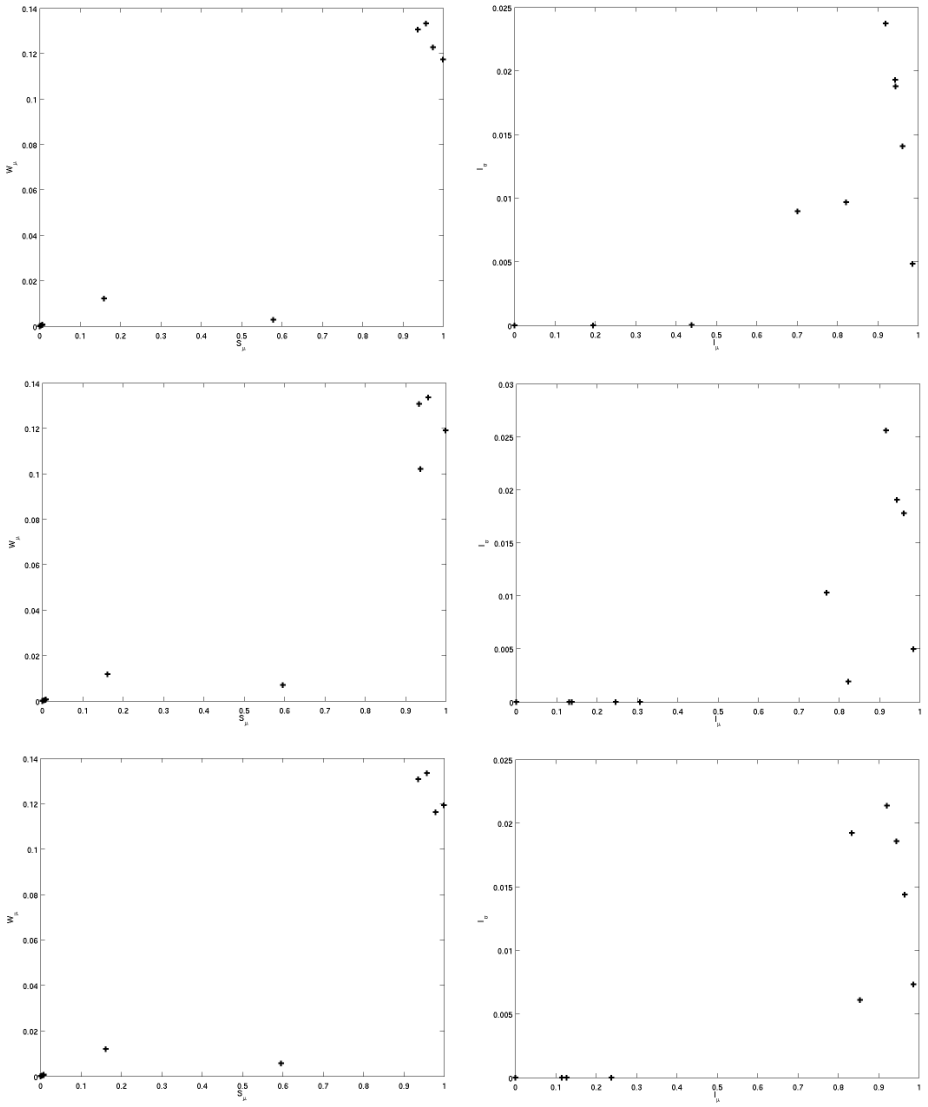

Figure 4.5: Dynamical Distributions of totalistic rules in $\mathsf{R}_{\mathbb{A}\,0}^{80}$ (top),$\mathsf{R}_{\mathbb{A}\,1}^{80}$ (middle), and $\mathsf{R}_{\mathbb{A}\,2}^{80}$ (botton) using a Von Neumann neighbourhood.

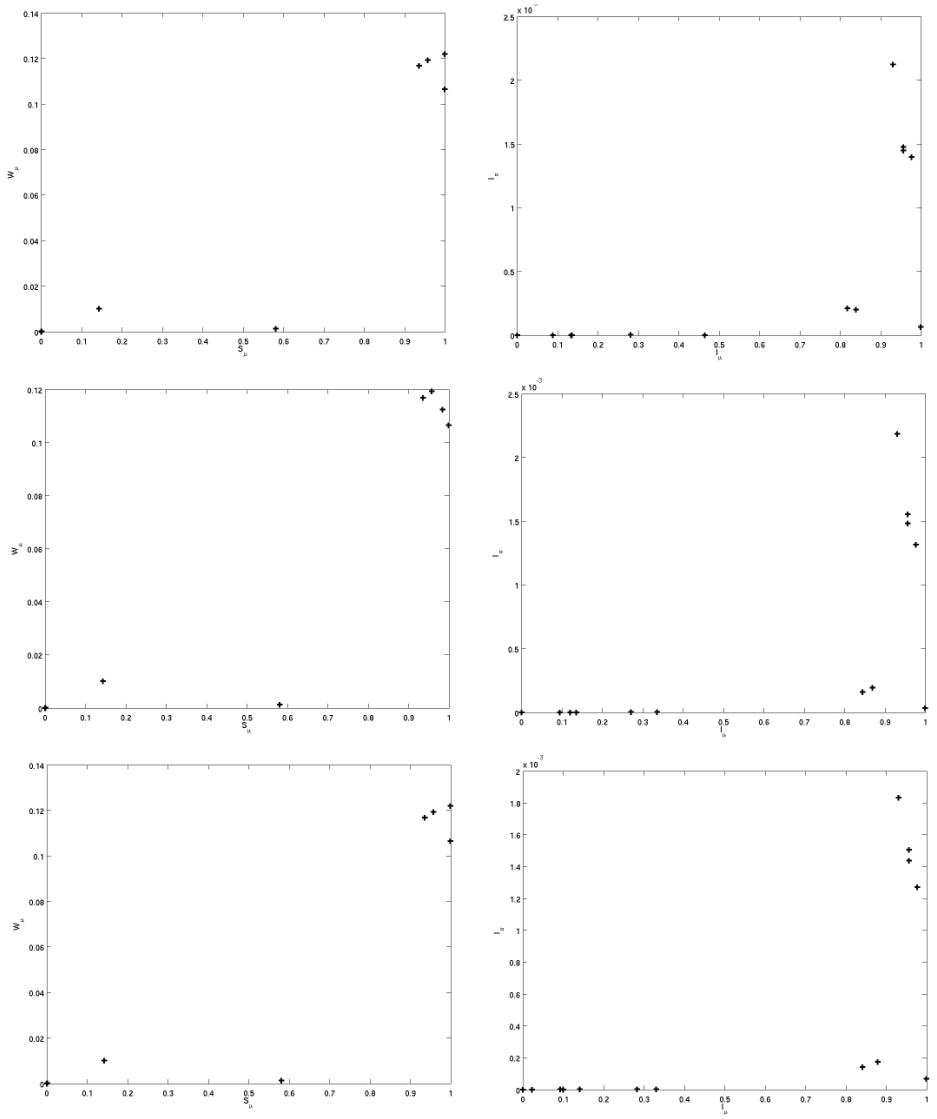

Figure 4.6: Dynamical Distributions of totalistic rules in $\mathsf{R}_{\mathsf{A}\,0}^{1280}$ (top),$\mathsf{R}_{\mathsf{A}\,1}^{1280}$ (middle), and $\mathsf{R}_{\mathsf{A}\,2}^{1280}$ (bottom) using a Von Neumann neighbourhood.

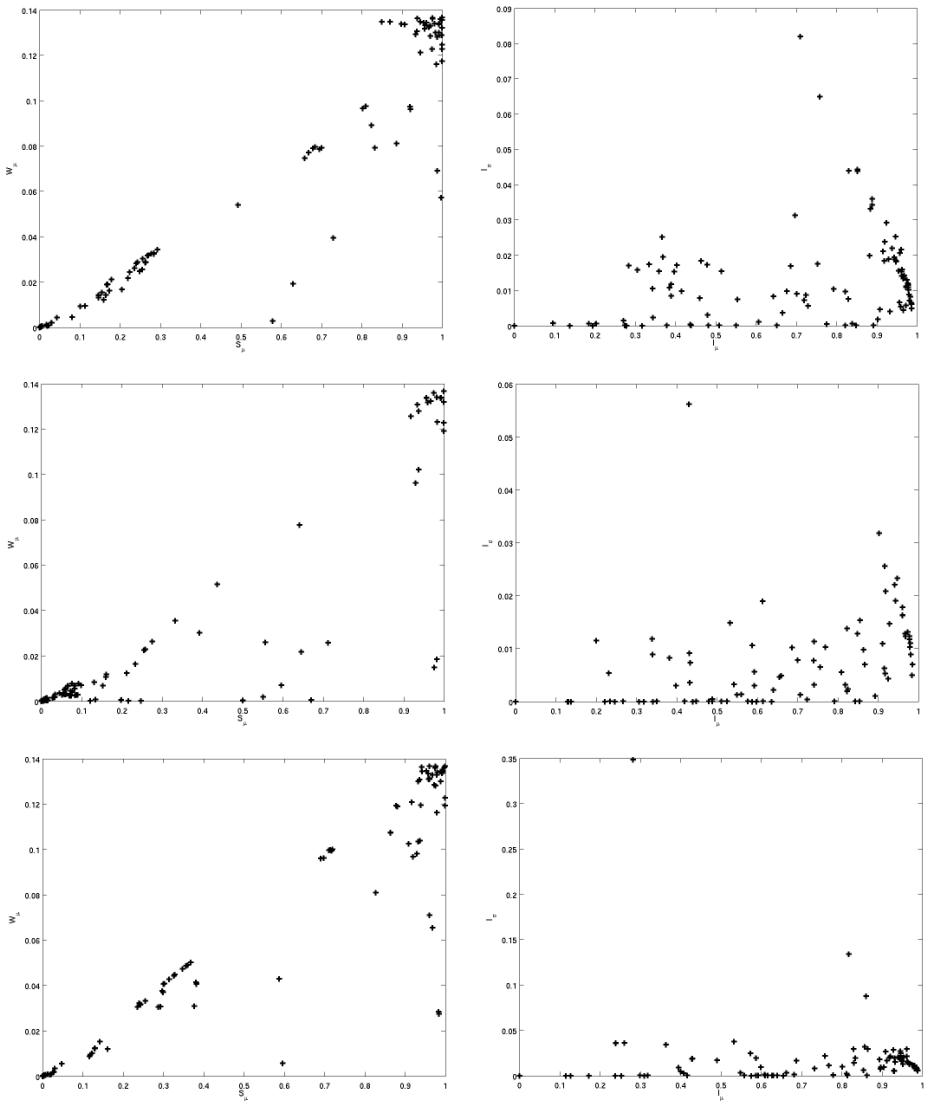

Figure 4.7: Dynamical Distributions of outer-totalistic rules in $\mathsf{R}_{\mathbb{A}\,0}^{80}$ (top),$\mathsf{R}_{\mathbb{A}\,1}^{80}$ (middle), and $\mathsf{R}_{\mathbb{A}\,2}^{80}$ (bottom) using a Von Neumann neighbourhood.

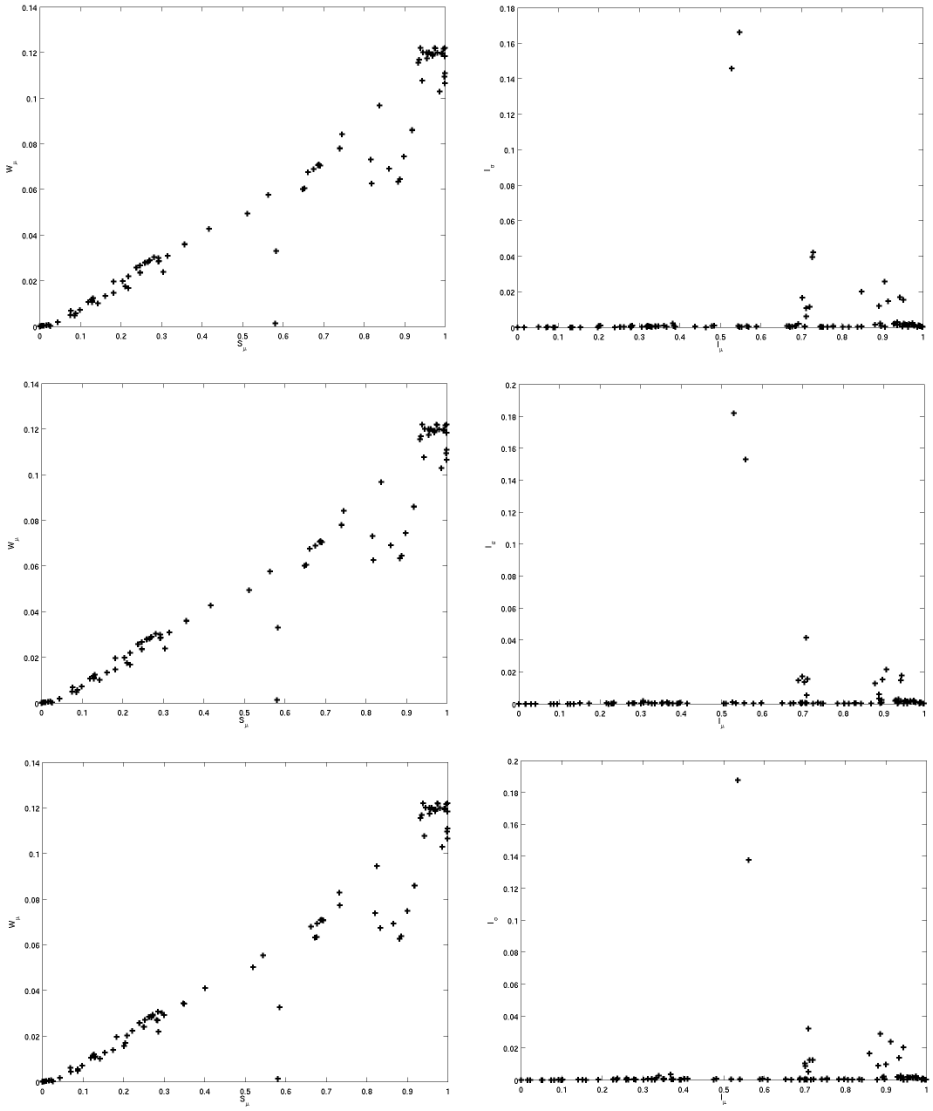

Figure 4.8: Dynamical Distributions of outer-totalistic rules in $\mathsf{R}_{\mathrm{A}\ {}^{1280}_{0}}$ (top),$\mathsf{R}_{\mathrm{A}\ {}^{1280}_{1}}$ (middle), and $\mathsf{R}_{\mathrm{A}\ {}^{1280}_{2}}$ (bottom) using a Von Neumann neighbourhood.

rules is shown in Figure 4.9. In this case, differences between rule spaces occur in similar regions (though the differences are not as extreme) in the outer-totalistic Von Neumann case with $n = 80$. More discussion on this will be presented in Section 4.4.

4.2.2 Configuration Transition Graph Properties

The configuration transition graph of a cellular automaton is a graph representation of the global dynamics of the automaton (analogous to phase portraits in continuous dynamics [68, 76]). As discussed in Section 2.3.2, there are several graph quantities that a useful for getting a picture the global dynamics; they are the mean transient length, the mean cycle length, and the G-density. Calculation of these quantities exactly is a computationally intensive task, for the purposes of my project these quantities were only taken for the sample sizes identical to the simulation cases as presented in Section 3.6.2.

It has been qualitatively shown in Section 4.2.1 that the Dynamical distribution of the rule space $R_{\mathbb{A}\,n}^{1280}$ seems to be minimally affected by changes in n (this is also validated quantitatively in Section 4.2.4). However, qualitatively there is still some variation in the average configuration transition graphs for these rules. This indicates that in this case rule exist which still have their global dynamics affected by topology even if the effect is not strong enough to case critical phenomena to occur.

Consider Figure 4.10, the distribution of transient lengths is effectively the same for any topological genus, but there is more noticeable difference in the cycle lengths. Rule spaces defined on a genus 1 topology seem to have more rules with smaller cycle lengths. This could represent Class II rules changing from a simple oscillator to a point attractor.

When considering only the totalistic rules in Figure 4.11, there is very little difference in configuration transition graphs. The mean cycle lengths are identical for all topologies. One rule varies in the mean transient length by a few hundred time-steps, however this may just represent a statistical outlier.

Just as in Section 4.2.1, the most interesting variation occurs when considering only the outer-totalistic rules. In Figure 4.12, there is very little change in the

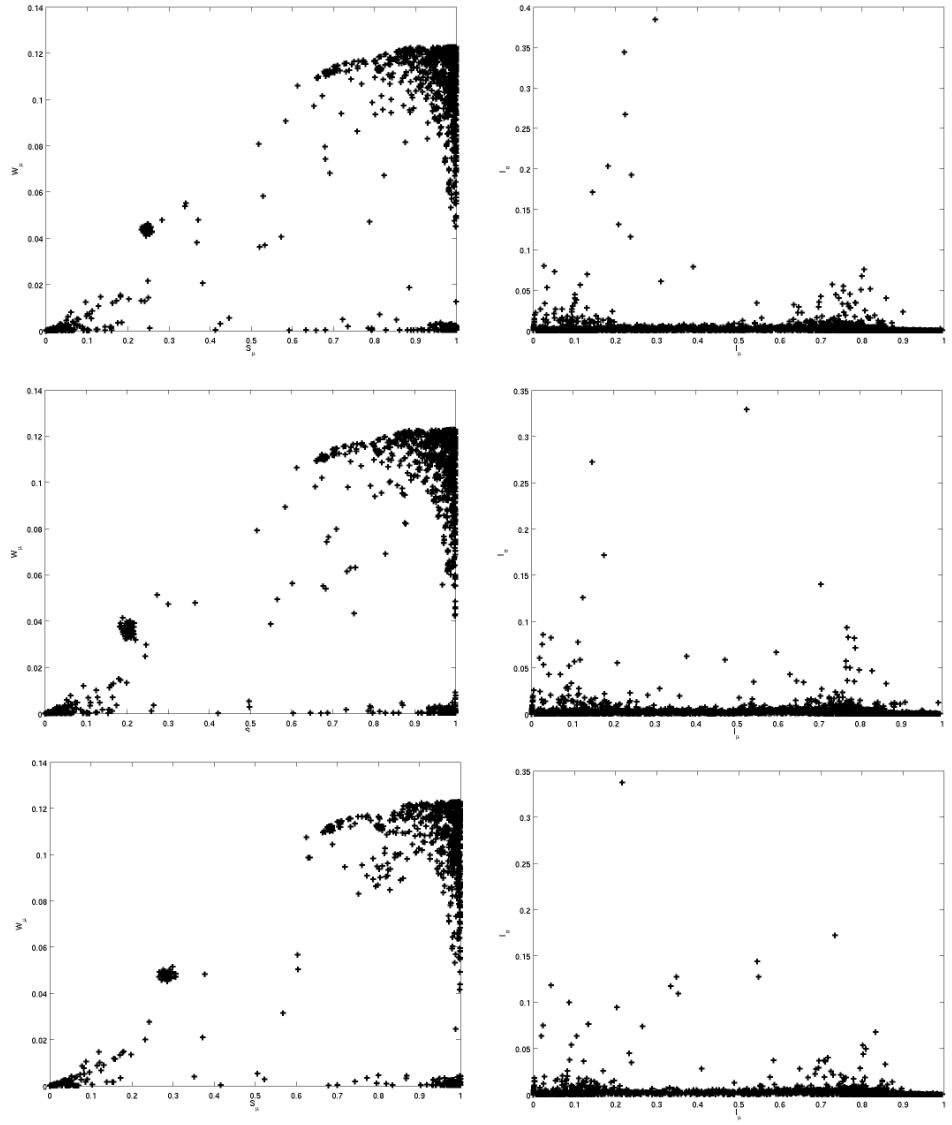

Figure 4.9: Dynamical Distributions of life rules (i.e., a special subset of outer-totalistic rules) in $\mathsf{R}_{\mathrm{A}\,0}^{1280}$ (top),$\mathsf{R}_{\mathrm{A}\,1}^{1280}$ (middle), and $\mathsf{R}_{\mathrm{A}\,2}^{1280}$ (bottom) using a Moore neighbourhood.

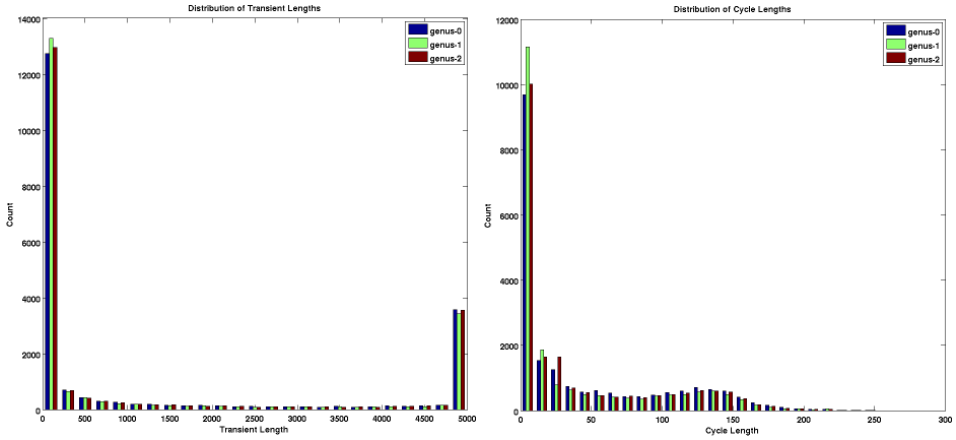

Figure 4.10: Distributions of average transient lengths and average attractor cycle lengths for rules in $R_{A\ 0}^{1280}$, $R_{A\ 1}^{1280}$, and $R_{A\ 2}^{1280}$ using a Von Neumann neighbourhood.

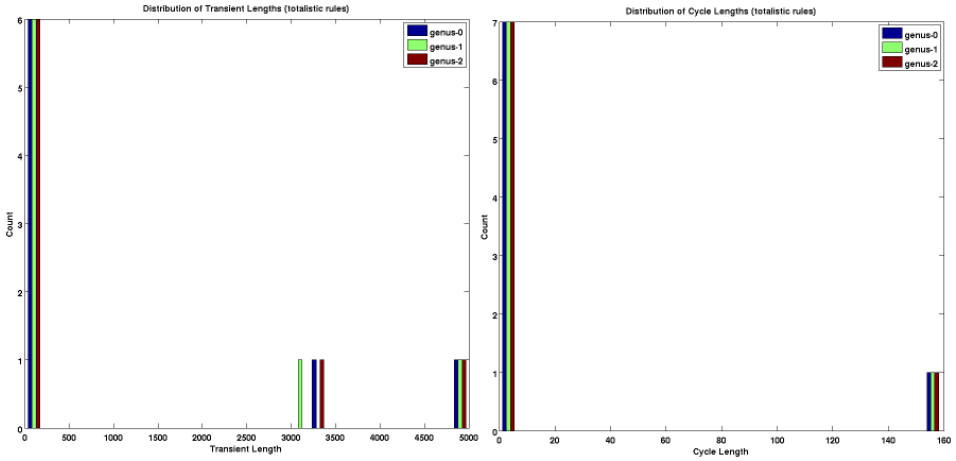

Figure 4.11: Distributions of average transient lengths and average attractor cycle lengths for totalistic rules in $R_{A\ 0}^{1280}$, $R_{A\ 1}^{1280}$, and $R_{A\ 2}^{1280}$ using a Von Neumann neighbourhood.

distribution of mean transient lengths, but there is quite a bit of variation in the mean cycle lengths. There are many more rules with very small mean cycle lengths for genus-1 topologies. The rules with cycle lengths in the range $[10; 40]$ are very much dominated by the genus-0 and genus-2 topologies. There is also another spike in the density of genus-1 rules around the cycle length of 120.

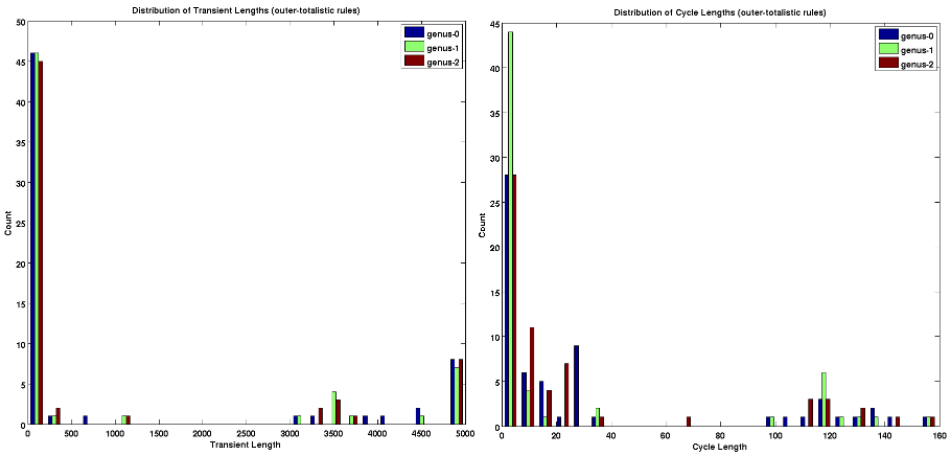

Figure 4.12: Distributions of average transient lengths and average attractor cycle lengths for outer-totalistic rules in $\mathsf{R}_{\mathbb{A}\ 0}^{1280}$, $\mathsf{R}_{\mathbb{A}\ 1}^{1280}$, and $\mathsf{R}_{\mathbb{A}\ 2}^{1280}$ using a Von Neumann neighbourhood.

The exact interpretation of these results in the light of Sections 4.2.1 and 4.2.4 will be developed more in Sections 4.3 and 4.4. The key point to consider at this stage is that topological genus may still impact the global dynamics of cellular automata even if the dynamical class of the automata remains the same.

4.2.3 Results of the E-test

The 3-sample E-test for distribution equality was applied to the results of each simulation case. Background on this E-statistic and the k-sample E-test is given in Section 3.7. The MATLAB$^{\mathbb{R}}$ code used to implement the test is provided in Appendix C.

Table 4.1: Results of the 3-sample E-test (with significance level □ = 0:05) for each simulation case.

Simulation case	p-value
VN_ALL_80	0:005
VN_TOT_80	0:99945
VN_OUTTOT_80	0:0003
VN_ALL_1280	0:6
VN_TOT_1280	1:0
VN_OUTTOT_1280	1:0
M_LIFE_1280	0:01

The 3-sample E-test computes the probability of a particular E-statistic value occurring under the assumption that the 3 samples come from the same distribution. For my project this is effectively computing the probability of the resulting E-statistic under the *null* hypothesis given in Equation (3.2).

A common statistical significance level for rejection of the *null* hypothesis is □ = 0:05. The number of bootstrap iterations required for the E-test to be accurate up to this level of significance is at least B = 100 (Recall from Section 3.7.3 that B must be selected such that (B + 1)□ 2 Z). Table 4.1 shows the resulting p-values for a significance level of □ = 0:05 and B = 1000.

4.2.4 Statistical Support for the Null Hypothesis

In Section 3.1, the formulation of the *null* hypothesis H_0 was presented. This hypothesis, which is defined in Equation (3.2), asserts that there is no difference the dynamical distributions of cellular automata rule spaces we defined on different topological genera. If strong evidence against H_0 exists then it is likely that changes on topological genus alone can have an effect on the global dynamics of cellular automata. It is important to note that finding evidence against H_0 is not sufficient information to indicate the existence of "critical phenomena", but rather indicates that such phenomena *could* exist.

A common method of rejecting the *null* hypothesis is to compare the p-value of the test statistic against a statistical significance level □. Typically if p < □,

then it can be said that sufficient evidence exists to reject H_0. At times this method of hypothesis rejection can be mis-used or mis-interpreted. Rather than be applying a binary decision of *reject*/*accept* the *null*, the following scheme is employed

□ p □ □: *No evidence against* H_0.

□ p < □: *Evidence against* H_0.

□ p < 0:1□: *Good evidence against* H_0.

□ p < 0:01□: *Strong evidence against* H_0.

The results of the E-test for the different simulation cases are shown in Table 4.1. Based on the above "levels-of-evidence" scheme and using □ = 0:05, there is no evidence against H_0 for rule spaces using a Von Neumann neighbourhood with a larger cell count (i.e., simulation cases VN_ALL_1280, VN_TOT_1280, VN_OUTTOT_1280) or for totalistic rules will small cell counts (i.e., simulation case VN_TOT_80). When considering the entire rule space, there is *good* evidence against H_0 for rule spaces using a Von Neumann neighbourhood with small cell counts (i.e., simulation case VN_ALL_80). The evidence against H_0 is strong for outer-totalistic rule spaces using a Von Neumann neighbourhood with small cell counts (i.e., VN_OUTTOT_80). There is also evidence against H_0 for outer-totalistic rules using a Moore neighbourhood with a larger cell count (i.e., M_LIFE_1280).

The "evidence" levels that are concluded from Table 4.1 seem to reflect closely the qualitative differences in dynamical distributions observed in Section 4.2.1. This provides quantitative support that changes in topological genus can effect the dynamical distribution of a graph cellular automata rule space. Certainly some of the observed regions of difference discussed in 4.2.1 do occur in regions of dynamical "phase-transition". Thus these statistically significant differences could be a source of "critical phenomena"; a more detailed exploration of these statistically significant regions is presented in Section 4.3.

4.3 Critical Phenomena Exemplars

The results from Section 4.2 indicate that the dynamical distribution of rule space can be effected (in some cases strongly) by the genus of the topology on which the cellular automata in the rule space are defined. In this section, it is shown that the effects of changes in topological genus extend to the individual cellular automaton. This effect at the individual level can be significant enough to cause a shift in the dynamical class of the automaton's evolution in the general sense, i.e., cause critical phenomena. It can even be demonstrated that these effects can exist in cases in which the effect on the dynamical distribution of rule space is near to non-existent.

Critical phenomena can be identified by significant changes in the rules entropy measures across different topologies. A difference in these measures is considered significant if the change is greater than the expected variation due to entropy calculation error (Section 4.1.2), and the change moves the rule into a different "region of dynamics" as discussed in Section 3.6.1. Using this approach, examples rules which undergo critical phenomena were identified in all simulation cases (excluding the totalistic cases).

Provided the local neighbourhood size is a large enough proportion of the total cellular automata cell count, the change in the dynamical distribution of rule space is (statistically) significant; thus the existence of critical phenomena is likely. This is the scenario for the simulation cases VN_ALL_80 and VN_OUTTOT_80. It comes as no surprise that critical phenomena examples are easy to find in both cases.

The nature of the critical phenomena varies from rule to rule, but in every observation that has been made thus far[1], one topological genus tends cause a change in dynamics for that rule whereas the same dynamics is observed with the other two genera. This can be demonstrated with rules 5738, and 18018 (Wolfram code).

The critical phenomena observed for the outer-totalistic rule 5738 is a change from simple oscillatory dynamics (i.e., Class II) when defined on a genus 0 or genus 1 topology to chaotic dynamics (i.e., Class III) when defined on a genus

[1] It should be pointed out that not all occurrences of critical phenomena have been analysed.

2 topology. However the non-totalistic rule 18018 exhibits chaotic dynamics on genus 0 and genus 2 topologies, but becomes oscillatory when defined on a genus 1 topology. This can be shown quantitatively by looking at the shift in the entropy measurements for this rule (Figures 4.13 and 4.14) and qualitatively using sample space-time patterns[1](Figures 4.15 and 4.16). There are also cases in which it is the genus 0 topology which causes the unique dynamics (e.g., rule 7650), since it is not feasible to present all of these cases, the reader is referred to Appendix D which contains a larger (but by no means comprehensive) listing of interesting entropy shift plots.

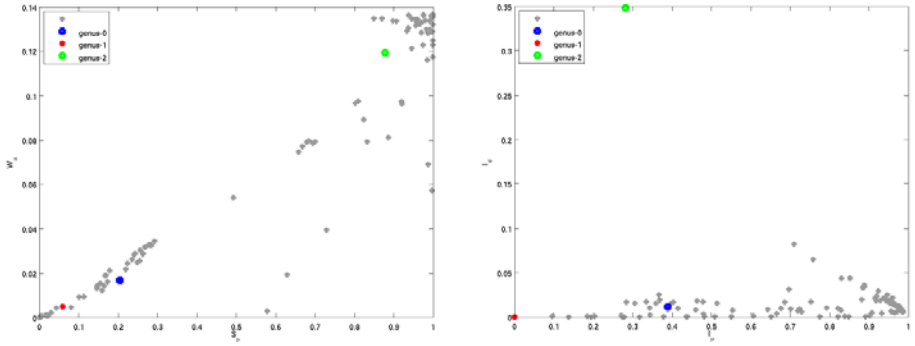

Figure 4.13: Entropy Measure shift observed in the outer-totalistic rule 5738 (defined on a Von Neumann neighbourhood).

It would be reasonable to conjecture that a cellular automata rule space with a small local neighbourhood relative to the automata cell count cannot exhibit critical phenomena since the dynamical distribution of rule space remains effectively unchanged. However, to the contrary, that examples of critical phenomena can be found in such cases (e.g., VN_ALL_1280, and M_LIFE_1280, but not VN_OUTTOT_1280), albeit such rules a far less common for these cases.

Some examples are the non-totalistic (Von Neumann Neighbourhood) rule 3986 (Wolfram code) and the outer-totalistic (Moore Neighbourhood) rule 7701

[1]Note that these space time patterns are □attened, and adjacent pixels do not represent actual neighbours.

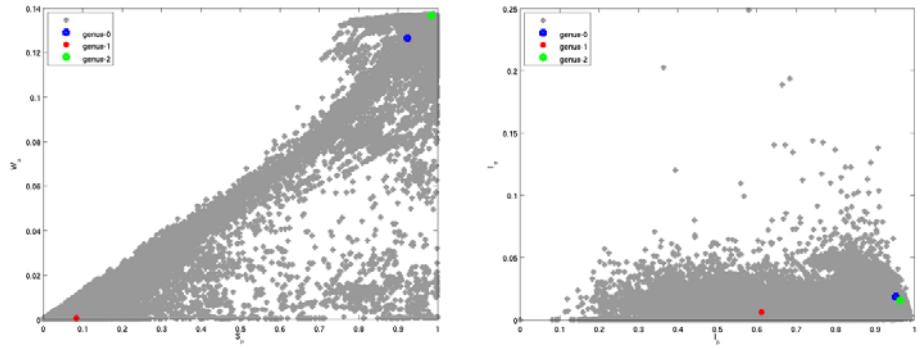

Figure 4.14: Entropy Measure shift observed in the non-totalistic rule 18018 (defined on a Von Neumann neighbourhood).

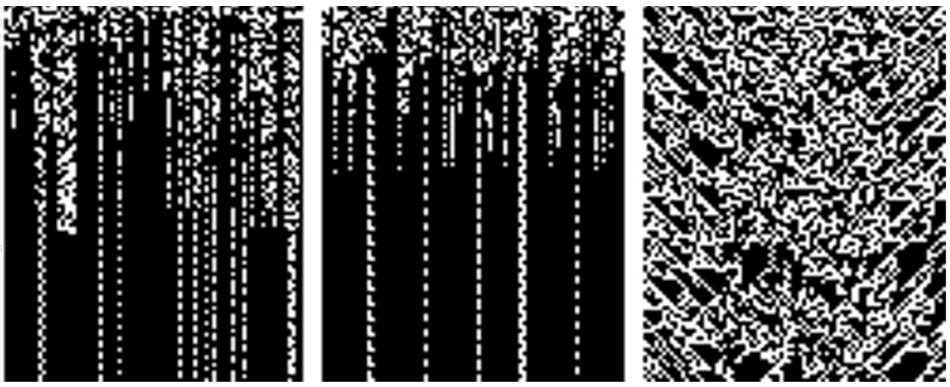

Figure 4.15: Critical phenomena observed in the rule 5738; Class II dynamics is observed on a genera 0 and 1 topologies (LEFT and CENTRE), whereas Class III dynamics occurs for genus 2 (RIGHT).

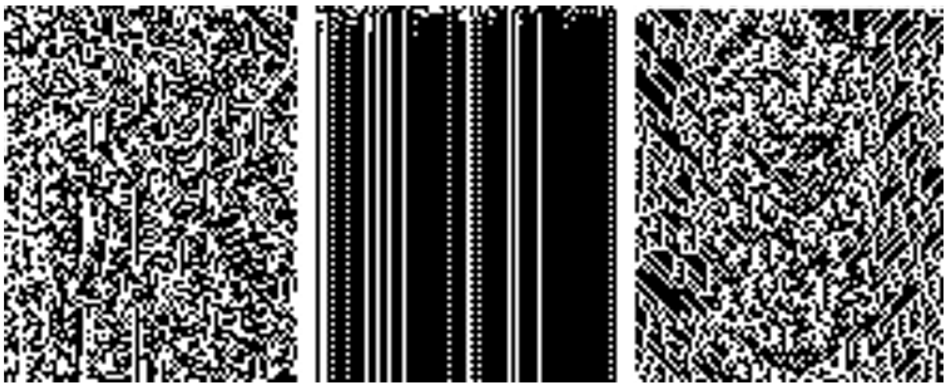

Figure 4.16: Critical phenomena observed in the rule 18018; Class III dynamics is observed on a genera 0 and 2 topologies (LEFT and RIGHT), whereas Class II dynamics occurs for genus 1 (CENTRE).

(Bays' notation). Just as for the 80 cell cases, the effect of the topological changes on individual rules is quite varied.

The non-totalistic rule 3986 exhibits simple oscillatory dynamics on a genus 1 or genus 2 topology, but becomes chaotic when defined on the genus 0 (Figure 4.17). Interestingly though, the rate of convergence to a simple oscillator is in general much slower for the genus 1 case (Figure 4.18). This pattern is actually common for the VN_ALL_1280 simulation case; this is discussed in more detail in Section 4.4.

Life rule 7701 is a particularly interesting example. This rule locally exhibits complex dynamics (Wolfram Class IV), yielding gliders and other complex structures. For example, the 22 period glider in Figure 4.19. Changing the topological genus on which the rule is defined seems to push the rule closer to either the chaotic or ordered side of the "edge-of-chaos".

This rule tends to be the most complex when defined on a genus 1 topology, whereas it has a tendency to explode into chaos on a genus 0 topology and tends to be "short-lived" on a genus 2 topology. This is reflected in the entropy shift pattern in Figure 4.20, in which the entropy signature moves from the chaotic

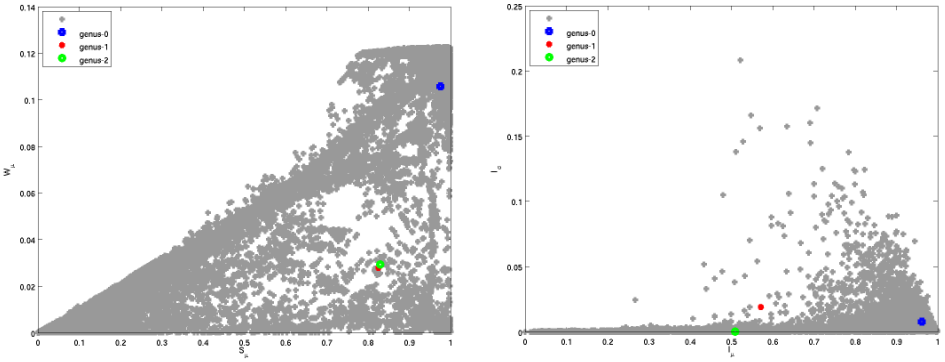

Figure 4.17: Entropy Measure shift observed in the non-totalistic rule 3986 (defined on a Von Neumann neighbourhood).

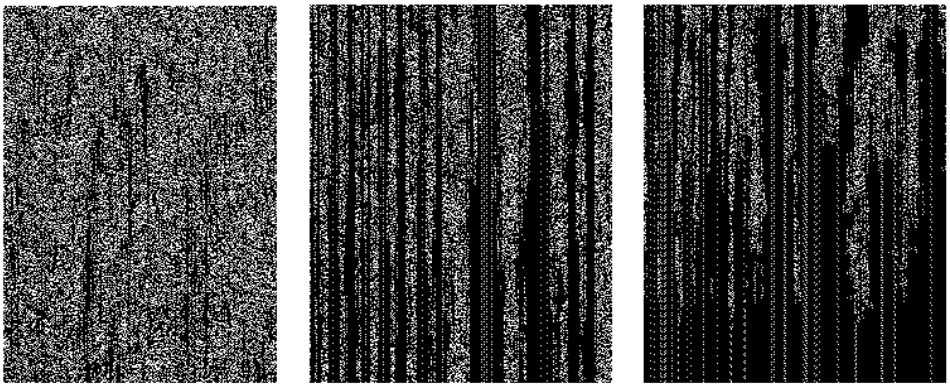

Figure 4.18: Critical phenomena observed in the rule 3986; Class II dynamics is observed on a genera 1 and 2 topologies (CENTRE and RIGHT), whereas Class III dynamics occurs for genus 0 (LEFT).

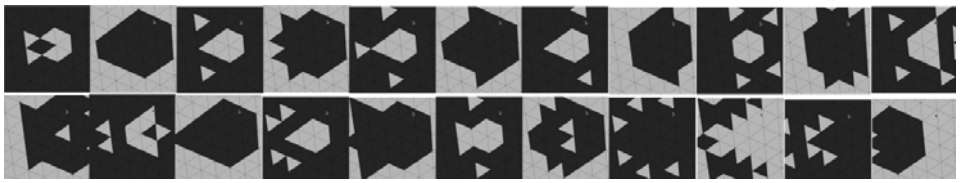

Figure 4.19: A glider produced by the outer-totalistic "Life" rule 7701 defined using the Moore neighbourhood. The glider has a period of 22 an proceeds forward two cells each period.

region to the oscillatory region as the genus increase. The long term input entropy is also shown in Figure 4.21. The genus 0 case remains close to 0:65, the genus 1 case starts to decrease but then suddenly (unexpectedly) increase demonstrating complex behaviour, finally the genus 2 case steadily decreases.

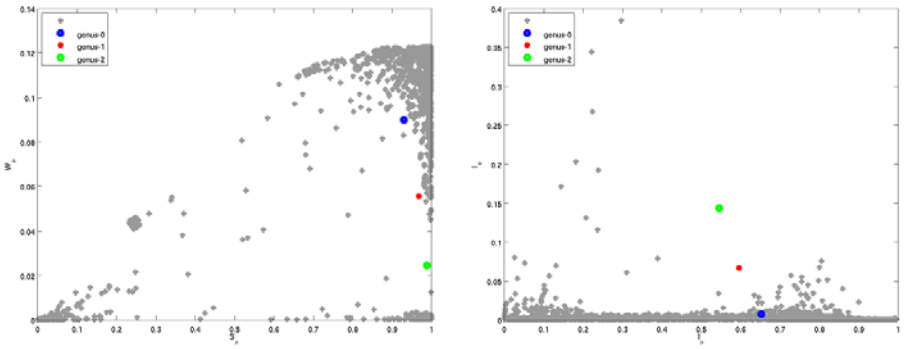

Figure 4.20: Entropy Measure shift observed in the outer-totalistic life rule 7701 (defined on a Moore neighbourhood).

Despite the unexpected discovery of critical phenomena in rule spaces that seem globally unaffected by topological genus, no examples of critical phenomena could be identified for purely totalistic rules. However, there do exist totalistic rules in which topology does affected the evolution of a fixed initial condition.

Totalistic rule 278 is a good example when only initial conditions consisting

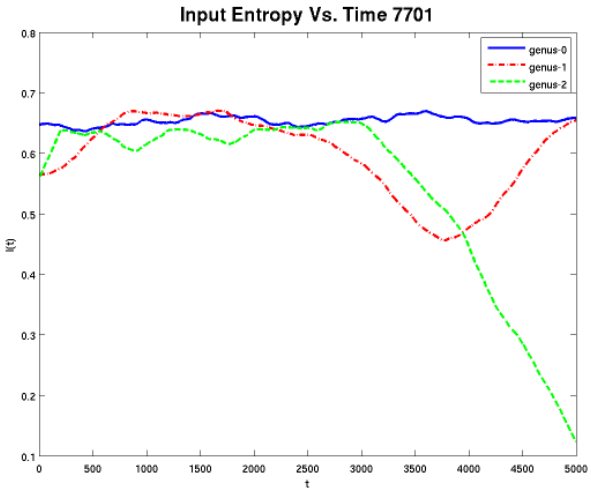

Figure 4.21: Input entropy over time for life rule 7701.

of a single "live" cell are considered. Rule 278 grows from a single point in a fractal-like pattern similar to the elementary cellular automaton *rule 90*. This evolution is shown in Figure 4.22.

The long-time evolution of rule 278 changes quite dramatically from chaotic behaviour (with an incredibly long attractor cycle) to a simple periodic pattern when the topology is changed from genus 0 to genus 1. Portions of the respective space-time plots are show in Figure 4.23.

Because we are only focusing on a single initial condition, it is easier to study the cause of this change in long-time behaviour. Consider the comparison of Figures 4.24 and 4.25. Both images sequences look similar until $\mathsf{t} = 16$ when the genus 0 has clearly produced a different structure. What is being seen here is the effect of the "pinch-points" (i.e., the regions just around the vertices of the icosahedron) on the genus 0 topology. Under normal "flat" evolution (i.e., genus 1) the self-replicating "stars" will always expand in a regular pattern until the periodic boundary is reached, and all "stars" collide in a synchronised fashion (i.e., two colliding "stars" always touch on time t such that $\mathsf{t} \mod 4 = 0$) thus spawning a new set of initial condition clones at regularly spaced locations. However

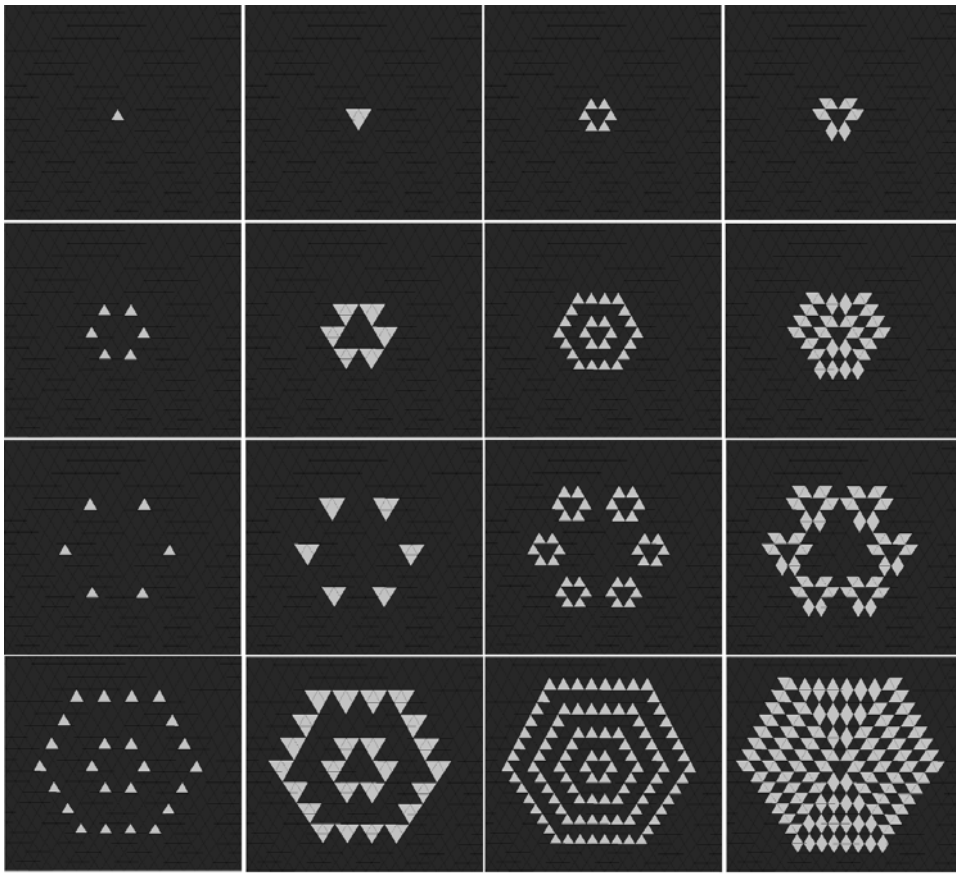

Figure 4.22: Fractal-like evolution of totalistic rule 278 from a initial point.

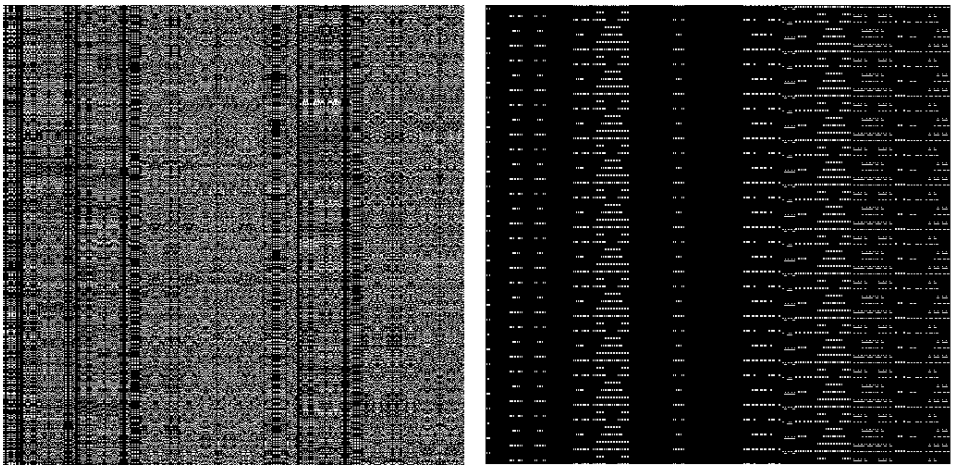

Figure 4.23: Space-time patterns for rule 278 from initial "point" contition for genus 0 (left) and genus 1 (right).

with the sphere (i.e., genus 2) the "stars" move slightly closer together around a pinch-point, causing two stars to collide "out-of-sequence" and destroying the repeating pattern. This phenomenon cannot be avoided as it is impossible to "flatten" a sphere (See Ventrella [57]).

The set of examples that have been shown in this section demonstrate that topological changes that preserve the homogeneity of cellular automata neighbourhoods can still have an effect on individual rules. This effect can be a-typical, only affecting the dynamics under a very specific set of initial conditions (e.g., rule 278 with point initial conditions). For some rules the effect can be so strong that the general dynamical classification of the rule can completely change. The strong effects of this nature can occur irrespective of the general effect of the topology on the dynamical distribution of rule space. The examples given in this section along with the visualisation and quantitative results in Section 4.2 provide a basis for some discussion presented in the following section.

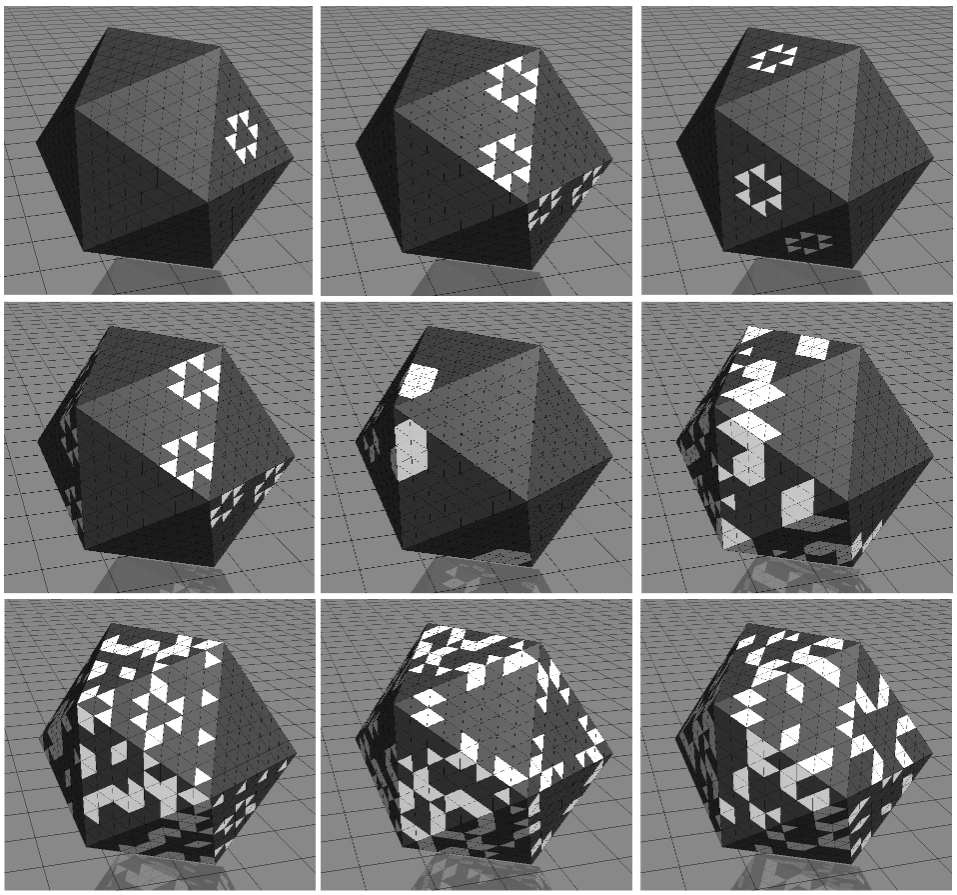

Figure 4.24: Evolution of rule 278 from a point on a genus 0 topology.

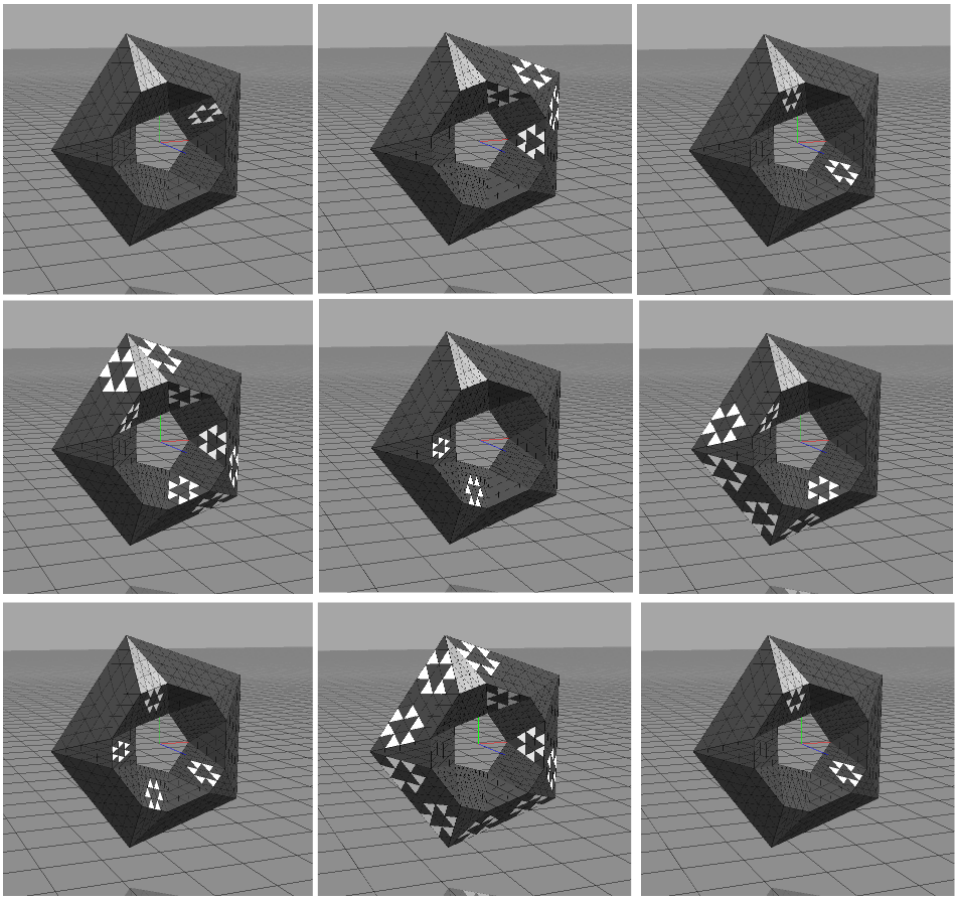

Figure 4.25: Evolution of rule 278 from a point on a genus 1 topology.

4.4 Discussion of Results

The topological perturbations that have been considered in this study are much more subtle that those investigated in the current literature [16, 32, 33, 42, 55]. By restricting mesh topologies to those that are equivalent to a surface mesh of a given genus it has been possible to look at how topological variation alone can effect cellular automata dynamics. Due to the preservation of neighbourhood homogeneity, a given graph cellular automaton may be defined on a topology of any genus while preserving local dynamics at any point[1].

The study of complex systems is concerned very much with the nature of how complex structures arise from simple interacting systems [64]. In Sections 4.2 and 4.3 it has been shown that the same set of simple interacting systems can in fact yield different dynamical systems in the general case (i.e., under random initial conditions) by operating over a different topology. The phenomena has been observed both from a global "rule-space" dynamical distribution perspective through to the perspective of individual rules dynamics and specific evolutions based on simple initial configurations.

The goal of this study has not been to locate general trends in the relationship between topology and cellular automata dynamics. However a few commonalities have been found in the types of effects that do occur; some of which will be discussed here.

4.4.1 Differences in Dynamical Distributions

The very existence of differences in dynamical distributions is quite significant. It shows that it is possible for a cellular automata rule space to have a higher density of a particular dynamical class of cellular automata when defined on one topology over another. It logically follows that it is possible to maximise the density of a particular dynamical class by variation of topological genus alone. The significance of this effect has been shown to differ widely across rule spaces, of which this section provides some insight.

[1] Strictly speaking, this is only true for totalistic and outer-totalistic rules. Non-totalistic rules are not invariant to neighbour \reordering", which means relative orientation must also be kept consistent; This is not possible to preserve across topologies.

The dynamical distribution of a cellular automata rule-space underwent the most significant alterations when the cell count was small (i.e., the local neighbourhood size was large by comparison). The most dramatic example of this is the change in the dynamical distribution of rules in $R_{\mathbb{A}\,n}^{80}$ under a Von Neumann neighbourhood.

For these small cellular automata the area occupied by a single neighbourhood is a large fraction of the overall cellular space. As a result the maximum distance of a neighbourhood to itself and the number of adjacent neighbourhoods many be significantly different on average, even though the total number of neighbourhoods remains unchanged. In some cases, neighbourhoods are nearly adjacent to themselves, which results in cells to having a greater influence on their own future states. This could be a significant factor on the resulting changes in dynamical distributions as the topological genus is increased.

So one might speculate, that changes in topological genus can only affect small cellular automata or cellular automata with extremely large local neighbourhood areas[1]. This is reinforced by the near zero change in dynamical distribution of rules in $R_{\mathbb{A}\,n}^{1280}$ for the Von Neumann neighbourhood. However, a more significant change is observed in the dynamical distribution of outer-totalistic rules in $R_{\mathbb{A}\,n}^{1280}$ for the Moore neighbourhood. This shift is qualitatively similar (thought certainly not as statistically significant) to the change observed in the dynamical distributions of outer-totalistic rules in $R_{\mathbb{A}\,n}^{80}$ for the Von Neumann neighbourhood. So the magnitude of the effect topological changes have on the dynamical distribution does not seem to be linearly dependent on neighbourhood size.

Dynamical distributions of totalistic rule spaces were observed to be completely unaffected by changes in topological genus. Though what happens beyond genus 2 has not been investigated, it is reasonable to conjecture that the effect is similar. As a result, totalistic rules could be considered stable under topological variation. This is a useful result from an fault-tolerant applications perspective.

Applications for cellular automata are often most interested in a particular dynamical class of automata. For example, chaotic rules (i.e., Wolfram Class III) are of interest for encryption applications [76], whereas complex rules (i.e., Wolfram Class IV) are of interest for computer architecture applications. The

[1] Relative to total cell count.

findings of this study indicate that it may be feasible to optimise a rule space for a given general application area. Though more work needs to be done, this is a significant finding in the study of complex systems as it could enable more focus in modelling approaches.

The results also suggest (though by no means prove) an answer to the question posed by Ventrella [57], that is "Are there unique spherical Machines?". The very existence of a topology yielding a higher density of a certain dynamical class could imply that rules exist in that class which perform computation that can only be achieved on that topology[1] It might be possible to use the results from this study to identify such a unique machine.

4.4.2 Critical Phenomena

The differences in dynamical distributions for rule spaces are, generally, insignificant for cellular automata with larger cell counts. As a result, the existence of critical phenomena occurring for larger cell counts was surprising.

The nature of the critical phenomena is varied across cellular automata. Some rules, such as life rule 7701, seem to transition from chaotic to complex to ordered classes as the genus increased. Other rules, such as outer-totalistic rule 55166 remained in an oscillatory dynamical class but the rate of convergence decreased. And still others, such as life rule 4757, change dynamical class, but a clear correlation to topological genus is not readily identifiable.

Overall though, certain trends in critical phenomena were still observed. For example, consider the near identical entropy shifts observed in rules 1842,3890, 3986, and 19226. These rules all exhibit chaotic/complex behaviour on a genus 0 topology, but exhibit oscillatory/ordered behaviour on a genus 1 or genus 2 topology. Another example, which seemed to mostly occur for smaller cell counts, is the entropy shift observed in rules 18018,33022,54612,43966, and 21846. In this case, cellular automata dynamics are homogeneous on a genus 1 topology,

[1] Except by simulation by a Turing complete cellular automaton. Theoretically, all possible cellular automata can be simulated by rule 110. This is not what is meant by \ Unique spherical Machine".

but genus 0 and genus 2 either both exhibit chaotic or both exhibit oscillatory behaviour.

What features of cellular automata rules influence the types of critical phenomena that can occur? This question is beyond the scope of this study. However, identification of such features may result in other cellular automata classifications beyond that of dynamical class..

The existence of critical phenomena with rule spaces which do not undergo changes in the dynamical distribution across topologies indicates a form of conservation law in these cases. That is, a rule may change dynamical class across topologies, but the total number of rules for each dynamical class remains constant[1] across topologies. For example, if a rule is found switches from chaotic dynamics on genus 0 to ordered dynamics on genus 1 then it must be that a set of rules exist in which the total of their dynamical changes overall achieves opposite.

What types of rule spaces have this property? This kind of conservation behaviour is observed in $\mathsf{R}_{\mathbb{A}\,1^{280}_{n}}$ using the Von Neumann neighbourhood. We can not speculate on any set of conditions that many be sufficient for the existence of such conservation behaviour, but a necessary condition could be a small local neighbourhood relative to the cell count.

4.4.3 Alterations in Specific Evolutions

It has been established in Section 4.4.1 that some cellular automata rules, such as totalistic rules, are stable under topological variations. However, even for these types of rules specific evolutions of a given initial condition may be altered dramatically. Though such cases have no relation to the dynamical classification of the rule itself, the existence of such case is still of interest.

The kind of effects observed are very much like those observed by Ventrella [57]. That is, alteration of a topology actually affects the regularity of the space and impacts how a particular pattern can grow. The most significant example of such alterations in regularity are the so called “pinch-points” of the genus 0 topology.

[1] At least approximately constant.

These pinch-points are the neighbourhoods around the vertices of the underlying icosahedron, such structures cannot be avoided [57].

In the case of totalistic rule 278 on the Von Neumann neighbourhood, the pinch-points greatly influence the evolution of the rule with an single point initial condition. When the growth of the fractal-like pattern reaches the pinch-points the boundary is contracted around these points, resulting in an "out-of-sequence" collision of structures which causes a chaotic pattern to emerge. On the genus 1 topology no such pinch-points exist, allowing the pattern to grow uniformly until it collides with itself, which in turn spawns a new point configuration and oscillatory dynamics begins.

Rules that produce gliders are also influenced by these irregularities. Ventrella observed that a glider produced by a rule similar to *Conway's Game of Life* displayed phenomena such as gliders spontaneously changing direction [57]. I have observed that gliders produced by life rule 7701 interact in different ways with the pinch-points depending on the "angle of impact". The glider is in some cases (e.g., head on collision) simply annihilated, but in other cases (e.g., offset collision) explodes into a complex/chaotic structure which is sometimes long-lived.

Different topologies also affect the number of potential collision points between two gliders. The trajectories of two perpendicular gliders cross each other twice on a genus 0 topology, once on a genus 1 topology, and on a genus 2 topology they need not cross at all. How these collision points effect the evolution of such glider rules in general could be of interest to cellular automata models of physical systems. This is another point of future work for which this this study forms a basis.

4.5 Summary

In this chapter the results of the methods described in Chapter 3 have been presented. This included the entropy error analysis and the comparison of dynamical distributions of rule spaces across different topologies.

The entropy error analysis revealed that sample sizes had the greatest influence on error, rather than sample size as a fraction of configuration spaces. This

allowed the approximation of the expected error when comparing distributions for a given sample size. It also provided a means of trading of performance and accuracy efficiently. Fortunately reasonable approximations could be made for the entropy measures using samples sizes which made simulations computationally feasible.

The dynamical distributions for rule spaces of the different simulation cases were qualitatively compared using visualisation, and quantitatively compared using the E-test. In both methods, the evidence indicated that topology can significantly change the rule distribution when the cell count is small, but some change is still observed for larger cell counts, even more so for the Moore neighbourhood.

Examples of critical phenomena behaviour have been provided for every simulation case, except for the totalistic cases. Though the effects are varied in both degree and type, there are some trends in what dynamical class a given genus is likely to induce.

Some discussion and conjecture has been presented in this chapter. This will be linked to the research questions in order to draw final conclusions in the next and final chapter.

Chapter 5

Conclusions

The purpose of this project was to investigate how a cellular automata rule space can be affect by changes in the topology of the network of cells. Very little is understood about behaviour of cellular automata on arbitrary topologies, and there are very few researchers who have dedicated any time to this area [16]. This project makes a unique and original contribution to this relatively new area of complex systems theory.

Research in complex systems theory through the use of cellular automata has primarily been conducted with respect a regular lattice of cells [42, 57]. To produce realistic models of real world phenomena more complex structure must be dealt with. The desire is to be able to construct models that are stable under alterations in topology, thus a deeper understanding of the interactions between structure and dynamics is required [16]. Relationships between structure and dynamics may also help focus the modelling process, which is often non-trivial.

The research questions of this project relate to the observation of critical phenomena in cellular automata caused by changes in the topological genus of the underlying cell space. These questions are as follows,

- Can critical phenomena be caused by topological variations which preserve homogeneity of local neighbourhoods?
- Do critical phenomena occur often enough to cause a significant change in the distribution of cellular automata dynamical classes across the rule space? If so, which topologies are best for maximising the concentration of a particular class.

- Can other dynamical differences still be observed across topologies when no critical phenomena are observed?

5.1 Results in Context

In this section, the results and discussion presented in Chapter 4 are related back to the original research questions in order to extract conclusions on these questions.

Can critical phenomena be caused by topological variations which preserve homogeneity of local neighbourhoods?

This study revealed many instances of cellular automata in which critical phenomena occurs under such topological changes. In some cases this was extreme enough to observe three classes of dynamics across the three topologies. By the enforcement of homogeneous local neighbourhoods, we have shown that global structure of cellular automata can alter dynamics to the same level as the local structure. This is a unique finding of this study, as previous work has look at significantly modified the homogeneity of the local neighbourhoods [33**?**].

Do critical phenomena occur often enough to cause a significant change in the distribution of cellular automata dynamical classes across the rule space? If so, which topologies are best for maximising the concentration of a particular class.

Of the various cellular automata rule spaces we extensively simulated, there were several in which the changes in the distribution of entropy values were statistically significant across the different topological genera. It follows that the overall distributions of dynamical classes is also affected to a similar significance.

The most significantly affected rule spaces had either a larger local neighbourhood definition or a very small cell count. It is reasonable to conjecture from this that for a sufficiently large local neighbourhood, variations in topology are guaranteed to impact the distribution of cellular automata dynamical classes. Conversely, to maintain stability of the rule space under such topological changes the neighbourhood size must be limited. The statistical significance levels also indicate that the effect of the distribution increases in a super-linear fashion with respect to relative neighbourhood size. One particular exception to

this conjecture,however, is that of totalistic rules. The results a quite conclusive that the distributions of dynamical classes for totalistic rule spaces is constant. More work needs to be performed to determine more detailed relationships with the distribution of rules and the topological genus.

The second part of this research question relates to the identification of a topology which maximises the density of a given dynamical class. The observation of changes in cellular automata dynamical classes due to topological changes implies that for a given dynamical class there exists such a maximising topology for that class. Is this topology the same regardless of the size of the cellular automaton and local neighbourhood size? We have not been able to investigate this in detail. We did however observe that a genus 0 topology seemed to host more chaotic dynamics, genus 1 more complex dynamics and genus 2 more ordered dynamics. If this trend is in fact what occurs in the general case, then this does have implications for design of applications and more broadly for complex systems theory.

Can other dynamical differences still be observed across topologies when no critical phenomena are observed?

The results have also shown that a cellular automaton that, under topological variation, does not undergo a significant change in dynamics in the average case can still generate significantly different space time patterns for a certain fixed initial condition. This is demonstrated clearly in the evolution of *rule 278* from an single point initial configuration across a spherical (i.e., genus 0) or toroidal (i.e., genus 1) grid as shown in Section 4.3. Variations of glider behaviours due to irregularities caused by the spherical grid support the conjecture by Ventrella that there may exist cellular automata evolutions which are unique to a certain topology [57].

5.2 Implications

There are several implications to complex systems theory and applications coming directly from this study.

Cellular automata are typically classified by the average long time evolution from random initial conditions, this was the classification scheme proposed by Wolfram. The results of this study have shown that certain cellular automata change in class depending on topology, even though local neighbourhoods are preserved. Largely this impacts a cellular automatons stability. It may be necessary, as we learn more about the role of topology in dynamics of cellular automata, that an additional classification be applied which is based on the nature of the critical phenomena observed on average. An example of such a scheme in the context of neighbourhood preserving topological changes could be,

- Class A: The dynamical class of the cellular automaton is a constant across topologies.
- Class B: The dynamical class of the cellular automaton transitions from ordered to chaotic as the topological genus increases.
- Class C: The dynamical class of the cellular automaton transitions from chaotic to ordered as the topological genus increases.
- Class D: The dynamical class of the cellular automaton transitions unpredictably between chaotic,complex, and ordered as the topological genus increases.

The identification of topologies with a higher density of certain dynamical class has implications for applications. Applications which depend on a certain dynamical class have a richer rule space to deal with if the maximising topology is selected; an example of such an application is cryptography which requires chaotic cellular automata. In the case of chaotic dynamics this is likely to be the genus 0 topology. Furthermore, consider the observation that complex dynamics seemed to be more common in the toroidal case, does this have implications for theoretical models of physics which are based on cellular automata? Could our universe in fact be a torus? Interestingly, this is already a theory held by some cosmologists [48].

When considering cellular automata as models of computation, topology may impact an algorithm's performance in terms of speed and accuracy. The potential

existence of machines which are unique to a given topology could mean that a given problem has an optimised solution only within a given topology. How significant can the performance gain be? Could it provide insight into relationships between computational complexity classes?

5.3 Future Work

This project has only scratched the surface of the interaction of topology and cellular automata rule spaces. There are still many unanswered questions. The most interesting I have listed here for future work.

Do the kinds of critical phenomena observed for binary cellular automata in the triangular tessellation continue to occur in cellular automata with larger state spaces or on other tessellations (e.g., square, or hexagonal)? What about larger neighbourhoods?

Do limiting behaviours occur as the topological genus increases? In this study, only topological genera 0, 1, and 2 where investigated. What happens as the genus approaches the cell count?

Can some of the conjecture made in this thesis be verified analytically? This project has been very much empirical and exploratory. With more focused theory it may be possible to prove the existence, or even a method of finding the topology which maximises a given dynamical class.

What novel applications can take advantage of these topological effects? How can we use knowledge of how a cellular automaton changes with topology to design new models?

Is there merit in the addition of a classification scheme based on the stability of cellular automata under topological variations? This has been very much an after thought of this project, but it may actually provide insight into relationships between discrete dynamical systems.

In summary, we have provided an initial platform for many future projects in complex systems theory to build off. We have answered some questions, but it leaves us asking even more. Hopefully a few, if not all, of the above topics are subject to a study in the near future.

5.4 Concluding Remarks

In this thesis, an original study as been presented investigating the effect of topology on cellular automata rule spaces. The findings indicate that the changes in topological genus alone can significantly impact the long time evolution of a given cellular automaton. Not only this, but the distribution of dynamical classes can be altered for the rule space as a whole. This represents a unique and original contribution to the area of complex systems theory, with potential far reaching implications understanding of the complexity which occurs around us in real biological and physical systems.

Appendix A

Publication: An Efficient Algorithm for the Detection of Eden

We have developed a polynomial-time algorithm to approximately solve the Eden problem for graph cellular automata and exactly solve the Eden problem for regular 1-d cellular automata. This algorithm was applied in order to compute entropy values for error analysis as described in Section 3.5.1. Aside from the immediate usage in this project, the Neighbourhood Elimination method has proven to been a very efficient method for solving the Eden problem with a significant reduction in the worst case computation time. The contents of this appendix is the original Journal paper manuscript which has been submitted and accepted to the Journal *Complex Systems* and is inprint to be published in a 2013 issue of the Journal.

An Efficient Algorithm for the Detection of Eden

David J. Warne*
School of Electrical Engineering and Computer Science,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia
Ross F. Hayward
School of Electrical Engineering and Computer Science,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia
Neil A. Kelson
High Performance Computing and Research Support,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia
Dann G. Mallet
School of Mathematical Sciences,
Queensland University of Technology,
Brisbane, Queensland 4001, Australia

In this paper, a polynomial time algorithm is presented for solving the Eden problem for graph cellular automata. The algorithm is based on our neighborhood elimination operation which removes local neighborhood configurations which cannot be used in a pre-image of the given configuration. This paper presents a detailed derivation of our algorithm from first principles, and a detailed complexity and accuracy analysis is also given. In the case of time complexity, it is shown that the average case time complexity of the algorithm is $\Theta(n^2)$, and the best and worst cases are $\Omega(n)$ and $O(n^3)$ respectively. This represents a vast improvement in the upper bound over current methods, without compromising average case performance.

1. Introduction

Cellular automata and more generally discrete dynamical systems are powerful tools for modeling of complex phenomena [14]. This includes applications from physics, biology, and computer science [1]. Some have even speculated that the study of cellular automata may lead to a Grand Unified Theory of everything [13].

The study of the global dynamics of cellular automata (i.e., the

*E-mail address: david.warne@qut.edu.au

study of automata configuration transition graphs) can provide unique insight into complex systems [20]. Efficient construction of a configuration transition graph typically requires a method to determine if the given configuration is located on a leaf node of this graph [17].

This problem is known as the Eden problem, and has been shown to be computationally intractable for d-dimensional systems when $d > 1$. This is reflected in the worst case computational complexity of algorithms that solve the Eden problem for higher dimensions (e.g., Wuensche's general reverse algorithm [19]).

We present a new algorithm for approximately solving the Eden problem for graph cellular automata (i.e., cellular automata on graphs [8, 7]); the most general form of deterministic cellular automata. Although there exist rare instances in which the algorithm will fail to identify the non-existence of a pre-image, this is made up for by it's asymptotic complexity class which is $O(n^3)$ for the worst case and $\Theta(n^2)$ for the average case. This provides a method which is more computationally feasible in the worst case than approaches based on Wuensche and Lesser's reverse algorithm [20] and Wuensche's general reverse algorithm [19] for the study of the global dynamics of higher dimensional discrete dynamical systems with potentially a large number of cells.

2. Background

2.1 Discrete dynamical systems

A regular cellular automaton can be defined as a lattice of finite state automata, typically referred to as cells or sites. A state transition function defines how a cell updates its state based on it's current state and the state of it's neighbors. Cells update synchronously in discrete time intervals. The sequence of all cell states at a given time is referred to as the automaton's configuration.

Random boolean networks are binary cellular automata with one critical difference; there is no requirement that cells be located on a regular lattice [19]. Instead, neighborhoods are constructed via a random wiring. This random wiring makes random boolean networks useful for theoretical biological models of genetic regulatory networks [5, 18].

Graph cellular automata (also referred to as Generalized automata networks [11]) are a generalization of both cellular automata and random boolean networks. For a graph cellular automaton, cell connectivity is defined by a connected graph. The class of graph cellular automata contains regular cellular automata and random boolean networks as sub-classes. Cellular automata and random boolean networks can be considered as discrete dynamical systems. Despite their simple construction, discrete dynamical systems have been shown to be

capable of very complex behavior [16, 15, 6]. Furthermore, computationally intractable, and formally undecidable problems relating to discrete dynamical systems have been shown to exist [3, 9].

2.2 The Eden problem

A particular problem of interest in the study of the global dynamics of discrete dynamical systems is the so-called Eden problem (also called the Predecessor existence problem [10, 2]). The Eden problem, attempts to determine, for a given automaton, if there exists a configuration (i.e., pre-image) that will evolve to the given configuration in the next time step. If the Eden problem is resolved to be false then the configuration is called a Garden-of-Eden configuration (i.e., it has no pre-image). Wuensche and Lesser studied the Eden problem in depth and developed a reverse algorithm for one dimensional regular cellular automata [20]. Wuensche further generalized this approach to the case of random boolean networks, which may also be applied to graph cellular automata [19, 17]. While Wuensche and Lesser's method performs very well for small cellular automata, this methods upper bound is $O(2^n)$ (as we will show in Section 5.1) which prevents exploration of large discrete dynamical systems.

For one-dimensional finite cellular automata the Eden problem is in P, however for multi-dimensional finite cellular automata the Eden problem has been shown to be NP-Complete [10]. Even certain variants of the Eden problem in one-dimension (such as the Constrained Eden problem [9]) have been shown to be NP-Complete. Assuming that $P \neq NP$, then there does not exist a polynomial time algorithm to solve the Eden problem for graph cellular automata.

If we assume $P \neq NP$, then a complete solution to the Eden problem for graph cellular automata is computationally intractable. However, this does not exclude the possibility of a good solution (i.e., one that can identify most Garden-of-Eden configurations) being achievable in polynomial time. In this paper, we present an algorithm which provides a good solution to the Eden problem for graph cellular automata in cubic time. By solving the problem for graph cellular automata we, by extension, solve the problem for regular cellular automata and random boolean networks. Furthermore, we can show that our algorithm solves the Eden problem exactly when the topology of the graph cellular automaton is equivalent to a one dimensional finite cellular automaton with periodic boundary conditions.

2.3 Formal definition of graph cellular automata

In this section, we provide a formal definition of graph cellular automata. Our formalism is based heavily on the work of Fates [4], Marr et al. [8, 7], and Tomassini [11].

We consider a graph cellular automaton to be defined as a 4-tuple consisting of a connected graph, a set of states, a set of neighborhood mappings and a set of state transition functions. This is given formally in Definition 1.

Definition 1. Let $A = (G, \Sigma, U, \Gamma)$ define a graph cellular automaton, where $G = (V, E)$ is a graph with vertices $V \subset \mathbb{Z}$ and edges $E \subseteq V \times V$, Σ is a finite set of symbols referred to as the alphabet, $U = \{h_i : i \in V\}$ is the set of neighborhoods $h_i = \{i\} \cup \{j : (i, j) \in E \vee (j, i) \in E\}$, and $\Gamma = \{g_i : i \in V\}$ is the set of all state transition functions $g_i : \Sigma^{|h_i|} \to \Sigma$.

In Definition 1, the vertices of the graph G represent the cells of the automaton A. Note that the neighborhood, h_i, of each cell, i, is effectively the set of cells that are connected to cell i via the set of edges E including i itself[1].

At any time t each cell is associated with a state σ. For this we define the mapping in Definition 2. From this we can construct the global configuration of the automaton in Definition 3.

Definition 2. Let $C : V \to \Sigma$ be a mapping from a cell $i \in V$ to a state $\sigma \in \Sigma$ such that $C^t(i)$ represents the state of cell i at time t. Let $C^t(h_i) \in \Sigma^{|h_i|}$ be the neighborhood configuration of i.

Definition 3. Let $\varphi^t = \{C^t(i) : i \in V\}$ be the configuration of the automaton A at time t. $\varphi^t \in \Phi$ where Φ is the set of all possible configurations of A.

Finally we define the evolution of a graph cellular automaton as the sequence of configurations generated by repeated synchronous application of the local state transition functions. This is given as a recurrence relation expressed in terms of the global configuration transition function. This is given in Definition 4.

Definition 4. Let the recurrence relation $\varphi^{t+1} = f(\varphi^t), t \geq 0$ be the evolution of A, where $f : \Phi \to \Phi$ is the global configuration transition function $f(\varphi^t) = \{(\varphi^t, \varphi^{t+1}) : \varphi^t = \{C^t(i) : i \in V\} \wedge \varphi^{t+1} = \{g_i(C^t(h_i)) : i \in V\}\}$.

We can now define formally an instance of the Eden problem.

Definition 5. Problem: The Eden problem (eden).
Instance: A graph cellular automaton A and a configuration $\varphi \in \Sigma^{|V|}$.
Question: Does there exist an initial configuration φ^0 such that $\varphi = f(\varphi^0)$ under the evolution of A?

[1] Note that the construction of h_i in Definition 1 assumes an undirected graph, the definition for a directed graph would be $h_i = \{i\} \cup \{j : (j, i) \in E\}$.

In Section 3, we will rely on the formalism given in this section to derive a polynomial time algorithm which provides the solution to eden(A,φ) in all but rare circumstances.

3. The algorithm

In this section, we present a detailed derivation of our Eden detection algorithm, denoted by eden-det (A,φ). There is a number of steps involved in this derivation. Firstly, some new mathematical constructions are defined. Then the fundamental operation of eden-det (A,φ), the neighborhood elimination operation, denoted by nh-elim(A,H), is derived. After presenting nh-elim(A,H) a simple Eden detection algorithm is provided, denoted by s-eden-det (A,φ). Using s-eden-det (A,φ) as a starting point we then derive a two phase construction of eden-det (A,φ).

3.1 Preliminaries

The graph cellular automata formalism given in Section 2.3 is not quite sufficient for us to express our Eden detection algorithm clearly. In this section, we present the definitions and notations that form the mathematical foundations of the algorithm. All definitions, notations, and theorems in this section assume the formalism in Section 2.3 to be given, and hence symbols used from Section 2.3 will not be re-defined.

We will assume, without loss of generality, that $\forall i \in V, |h_i| = k$. This is done purely for notational convenience. All of the concepts applied in the construction of our algorithm can be extended trivially to non-uniform $|h_i|$. Note that we do not assume a uniform update rule across all cells $\forall i, j \in V, g_i = g_j$.

To describe our algorithm, we need a method of consistently referring to a specific neighborhood configuration (see Section 3.2). The notation for this reference is given in the following definition.

Definition 6. Assume that some ordering scheme has been applied to the set of all neighborhood configurations Σ^k. Subject to this ordering, the nth neighborhood configuration is denoted by $\psi_n \in \Sigma^k$.

Note that the actual ordering scheme is arbitrary, all we require is an index into the possible neighborhood configuration space. For our implementation we simply map each configuration to its raw binary representation.

It is necessary for us to specify a set that contains all the cells that join adjacent neighborhoods. We refer to this set using the notation ${}^{i}\Xi^{j}$, and it is defined in Definition 7.

Definition 7. Let ${}^{i}\Xi^{j} = \{i\} \cup \{j\} \cup \{x : ((x,i) \in E \vee (i,x) \in E) \wedge ((x,j) \in E \vee (j,x) \in E)\}$ denote the boundary set of h_i and h_j. The

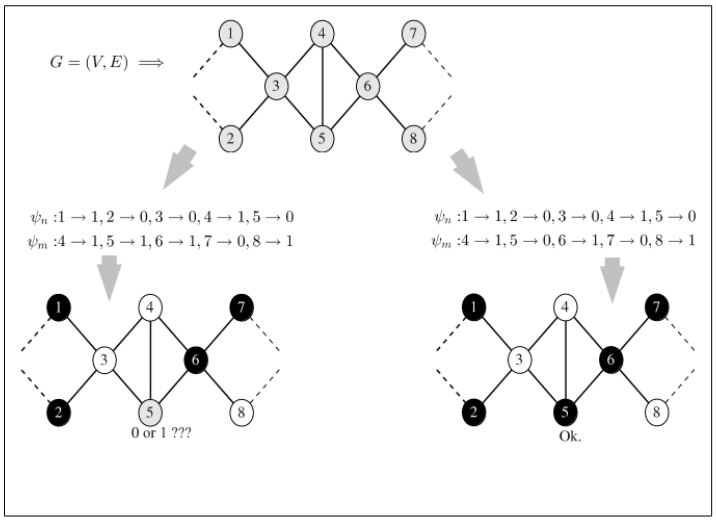

Figure 1. Example of $i\Box j$-consistency where $i = 3$, $j = 6$, and ${}^{i}\Xi^{j} = \{3\Box 4\Box 5\Box 6\}$. LEFT: ψ_n is not $i\Box j$-consistent with respect to ψ_m since they cause an inconsistent state in the boundary set (i.e., cell 5). RIGHT: A modification to ψ_m allows consistency across the boundary set, hence ψ_n is now $i\Box j$-consistent to ψ_m.

nth boundary cell, $x \in V$, is denoted by $x = {}^{i}\Xi^{j}_{n}$.

The basis of our algorithm is the detection and removal of neighborhood configurations which cannot exist in any pre-image of φ^t due to an inconsistency across boundary sets.

Definition 8. If there exists an initial configuration φ^0 such that $C^0(h_i) = \psi_n$ and $C^0(h_j) = \psi_m$, then ψ_n is said to be $i\Box j$-consistent with respect to ψ_m.

The concept of $i\Box j$-consistency is readily visualized as shown in Figure 1. However, it would be preferable if a direct method of evaluating the $i\Box j$-consistency of two neighborhood configurations could be found. The function we require is given in Definition 9.

Definition 9. Let $\theta_i^{{}^{i}\Xi^{j}} : \Sigma^k \to \Sigma^{|{}^{i}\Xi^{j}|}$ be a function which maps neighborhood configurations of h_i to the configuration of the boundary set ${}^{i}\Xi^{j}$. The function is defined as $\theta_i^{{}^{i}\Xi^{j}} = \{(\psi_n\Box\psi'_n) : \psi'_{n\Box s} = \psi_{n\Box q} \wedge y = h_{i\Box q} \wedge y = {}^{i}\Xi^{j}_{s} \wedge s \in [0\Box|{}^{i}\Xi^{j}|)\}$.

The definition of θ given in Definition 9 may seem strange, but it leads us into Theorem 1 which is a vital component of our algorithm.

Theorem 1. If $\theta_i^{\not\equiv j}(\psi_n) = \theta_j^{\not\equiv j}(\psi_m)$ then ψ_n is i-j-consistent with respect to ψ_m.

A formal proof is given in Appendix A.

3.2 Neighborhood elimination

We can now formulate the core operation of our Eden detection algorithm. This core operation we call neighborhood elimination and denote it as nh-elim(A,H). As the name may suggest, its function is to eliminate neighborhood configurations which cannot be a component of any pre-image of the automaton configuration in question.

To explain how we perform this operation, we first consider the matrix $H \in \{0,1\}^{|\Sigma|^k \times |V|}$ where

$$H_{i,j} = \begin{cases} 0, & \text{impossible for } \psi_i = C^0(h_j) \\ 1, & \text{otherwise} \end{cases}, \qquad (1)$$

It is important to note in Equation (1), that $H_{i,j} = 1$ should not be interpreted as $\psi_i = C^0(h_j)$ in at least one pre-image. Instead, $H_{i,j} = 1$ means we cannot yet determine if $\psi_i = C^0(h_j)$ or not. This is not the case for $H_{i,j} = 0$, which indicates that we have proven that there is no pre-image such that $\psi_i = C^0(h_j)$.

The algorithm can be described as follows: Consider the case in which we have already determined that $H_{i,j} = 0$ for specific i,j by techniques described in Section 3.3. If we start with an arbitrary cell neighborhood h_i then the column vector $H_{*,i}$ provides us with the neighborhood configurations still under consideration. If $H_{n,i} = 1$, but the neighborhood configuration ψ_n is not i-j-consistent with respect to any candidate configurations in one or more connected neighborhoods h_j, then ψ_n can be excluded from the realm of possible configurations for h_i as at least one boundary cell state cannot be satisfied consistently. By updating $H_{*,i}$ this will affect the validity of other configurations, so we repeat the process for every neighborhood.

Theorem 1 provides us with a comparison operation for testing the i-j-consistency of two neighborhood configurations. With the function $\theta_i^{\not\equiv j}$ as defined in Section 3.1, we arrive at nh-elim(A,H) (i.e., Algorithm 1).

One step of nh-elim(A,H) is shown in Figure 2 displaying contents of the data structures Θ_i,Θ_j, and ζ_i along with the effect on the state of H. It should be noted that although the example in Figure 2 is for a small 1-d cellular automaton with k = 3, nh-elim(A,H) is general enough to operate on graph cellular automata.

One particularly useful property of nh-elim(A,H) is that the number of zero elements in H can never decrease. Therefore, repeating nh-elim(A,H) on H in an iterative fashion will eventually result in

Algorithm 1 nh-elim(A,H): Neighborhood elimination.

```
for all i ∈ V do
  for all j ∈ h_i − {i} do
    Θ_i ← { x : x ∈ θ_i^{i≡j}(ψ_p) ∧ H_{p,i} = 1 }
    Θ_j ← { x : x ∈ θ_i^{i≡j}(ψ_q) ∧ H_{q,j} = 1 }
    ζ_i ← {x : x ∈ Θ_i ∧ x ∉ Θ_j}
    ∀p H_{p,i} ← 0 if θ_i^{i≡j}(ψ_p) ∈ ζ_i
  end for
end for
```

$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

$$H = \begin{bmatrix} & k & k+1 & \\ \dots & 0 & 0 & \dots \\ \dots & 1 & 0 & \dots \\ \dots & 0 & 1 & \dots \\ \dots & 1 & 1 & \dots \\ \dots & 0 & 1 & \dots \\ \dots & 0 & 0 & \dots \\ \dots & 0 & 0 & \dots \\ \dots & 0 & 0 & \dots \end{bmatrix}$$

$$\Theta_i \leftarrow \left\{ x : x \in \theta_i^{i\equiv j}(\psi_p) \wedge H_{p,i} = 1 \right\}$$

$$\Theta_i = \{01, 11\}$$

$$\Theta_j \leftarrow \left\{ x : x \in \theta_i^{i\equiv j}(\psi_q) \wedge H_{q,j} = 1 \right\}$$

$$\Theta_j = \{01, 10\}$$

$$\zeta_i \leftarrow \{x : x \in \Theta_i \wedge x \notin \Theta_j\}$$

$$\zeta_i = \{11\}$$

$$H' = \begin{bmatrix} \dots & 0 & 0 & \dots \\ \dots & 1 & 0 & \dots \\ \dots & 0 & 1 & \dots \\ \dots & 0 & 1 & \dots \\ \dots & 0 & 1 & \dots \\ \dots & 0 & 0 & \dots \\ \dots & 0 & 0 & \dots \\ \dots & 0 & 0 & \dots \end{bmatrix}$$

$$H_{p,i} \leftarrow 0 \ \forall p, (\theta_i^{i\equiv j} \in \zeta_i)$$

$$H_{4,k} = 0$$

Figure 2. One Step of nh-elim(A,H). The upper left matrix is H at the start of the iteration, after the iteration is completed $H_{4,k}$ is set to 0, resulting in the lower left matrix H'.

an array H in which only configurations i-consistent with respect to all neighbors are candidates for pre-image construction. This property also enables us to put an upper bound on the number of iterations required, which aids us in our complexity analysis (see Section 4).

3.3 Garden of Eden detection

In this section, we will describe our Eden detection algorithm (eden-det (A,φ)) in full. Throughout this description we rely heavily on the formalism in Section 2.3 and Section 3.1.

So far we have assumed that H is not all ones or all zeros, but we

have not mentioned how H is initialized. If we are given an instance of eden(A,φ), we can prove the impossibility of some neighborhood configurations explicitly by using φ and the state transition functions $g_i \in \Gamma$, that is,

$$H_{i,j} = \begin{cases} 0 & g_j(\psi_i) \neq \varphi_j \\ 1 & g_j(\psi_i) = \varphi_j \end{cases} \quad (2)$$

In Section 3.2, it was stated for nh-elim(A,H) (Algorithm 1) that the number of zeros in H can never decrease. Therefore, repeated invoking of nh-elim(A,H) is guaranteed to converge to a steady state.

Once H is initialized, we can repeatedly operate the neighborhood elimination algorithm on H. Clearly, if for any column $\forall i, H_{i,j} = 0$ during a iteration, then there is no possible ψ_i that can be selected for h_j in any pre-image. Furthermore, the steady state that H will converge to in this case will be $\forall i,j, H_{i,j} = 0$. Therefore we can conclude that φ is a Garden of Eden configuration.

We might also assume that all Garden of Eden configurations will cause the condition, $\forall i, H_{i,j} = 0$. Therefore, we could simply iterate until a steady state is reached and then look at the elements in H for any non-zero elements. This leads us to derive our initial algorithm for Eden detection, which we call simple eden detection and denote as s-eden-det(A,φ) (Algorithm 2).

Algorithm 2 s-eden-det(A,φ): Simple Eden detection.

```
∀i,j, H_{i,j} ← 0 if i,j, (g_j(ψ_i) ≠ φ_j)
∀i,j, H_{i,j} ← 1 if i,j, (g_j(ψ_i) = φ_j)
while H ≠ H' do
  H' ← H
  H ← nh-elim(A, H')
end while
if ∀i,j, H_{i,j} = 0 then
  GoE ← true
else
  GoE ← false
end if
return GoE
```

Unfortunately, s-eden-det(A,φ) is not quite complete[2]. It can be shown that $\forall i, H_{i,j} = 0$ is a sufficient but not necessary condition of Eden. It is possible for cells within a cycle of G to have i,j-consistent neighbors, but there does not exist a combination of possible neighborhood configurations that can form a consistent chain. Figure 3 gives an example of such a case, note that for a 1-d cellular automaton the topology graph G contains one cycle which includes all cells. Clearly, more processing is required once s-eden-det(A,φ) has converged and

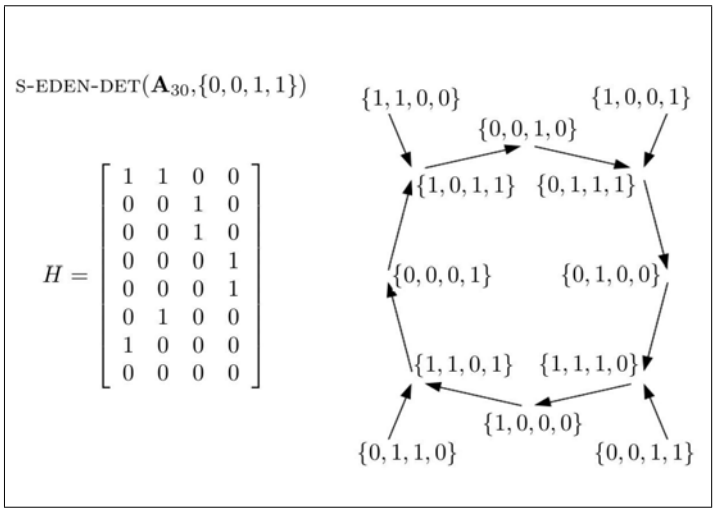

Figure 3. Counter example for s-eden-det (A, φ) (Algorithm 2). LEFT: The resulting non-zero steady state of H where A_{30} is the elementary cellular automaton rule 30 (according to Wolfram's numbering scheme [16]) with periodic boundary conditions and $|V| = 4$. RIGHT: The main configuration transition graph for A_{30}, clearly $\varphi = \{0, 0, 1, 1\}$ has no pre-image.

there does not exist a $j \in V$ such that $\forall i, H_{ij} = 0$.

If for any possible neighborhood configuration (i.e., $H_{ij} = 1$) we can construct at least one pre-image, then we can conclude that φ is not a Garden of Eden configuration. However, if it is found that a valid pre-image cannot be constructed with $C^0(h_j) = \psi_i$ then we can set $H_{ij} = 0$ and repeat s-eden-det (A, φ) until a new steady state is reached. This leads us to a second and more complete approach eden-det (A, H).

In practice, we locate each instance of $H_{ij} = 1$, and temporarily set $\forall k \in i, H_{kj} = 0$. This has the effect of assuming that $C^0(h_j) = \psi_i$. We then apply one iteration of nh-elim(A, H) ensuring that cell $j \in V$ is visited last, then we examine the state of H_{ij}. If $H_{ij} = 1$, then we have no reason to reject our assumption. Otherwise, our assumption is disproved via contradiction, so we set $H_{ij} = 0$ and repeat the loop in s-eden-det (A, φ). If none of the $H_{ij} = 1$ can be disproved, then it is reasonable to conclude that φ has at least one pre-image (We show in Section 6 that there are rare cases when this is an invalid conclusion).

We now have a two phase procedure. Phase 1, denoted by ph1(A, H)

[2] Hence the name simple Eden detection.

(Algorithm 3), is effectively the loop from s-eden-det (A, φ). Phase 2, denoted by ph2(A, H) (Algorithm 4), is the assumption testing process described in the preceding paragraph. These two phases are then combined to form our full Eden detection algorithm eden-det (A, φ) (Algorithm 5). A implementation of eden-det (A, φ) is provided as part of analysis software developed by Warne [12][3].

Algorithm 3 ph1(A, H): Eden detection phase 1.

```
while H ≠ H′ do
  H′ ← H
  H ← nh-elim(A, H′)
end while
```

Algorithm 4 ph2(A, H): Eden detection phase 2.

```
for all i ∈ V do
  for all j ∈ Σ^|k| do
    if H_ij = 1 then
      H^tmp ← H
      ∀s ∈ (s ≠ j) H^tmp_is ← 0
      H^tmp ← nh-elim(A, H^tmp)
      if H^tmp_ij = 0 then
        H_ij ← 0
        return
      end if
    end if
  end for
end for
```

Leaving the details to Section 4, we simply claim that eden-det (A, φ) is guaranteed to complete in polynomial time. More specifically, it can be shown to have a cubic worst case time efficiency. Furthermore, when eden-det (A, φ) returns GoE = false, then H encodes the complete set of pre-images to φ^t (except for rare cases when GoE = false is a false negative as shown in Section 6).

4. Time complexity analysis

In this section, we present the time complexity analysis for eden-det (A, φ) (Algorithm 5). We show that the number of operations for the best case is a linear function of the number of cells, the worst case is shown to be cubic, and the average case is shown to be quadratic.

[3] This software, called GCALab, is an command line analysis tool designed for parallel computation of graph cellular automata properties.

Algorithm 5 eden-det (A, φ): Eden detection.

```
∀i,j ∈ H_ij ← 0 if i ∈ j ∈ (g_j(ψ_i) ∉ φ_j)
∀i,j ∈ H_ij ← 1 if i ∈ j ∈ (g_j(ψ_i) = φ_j)
repeat
  call ph1(A,H)
  if ∀i,j ∈ H_ij = 0 then
    GoE ← true
    return GoE
  else
    call ph2(A,H)
    GoE ← false
  end if
until H′ = H
return GoE
```

Experimental results are also presented to reinforce theory with practice.

4.1 Time complexity of NH-ELIM($A \in H$)

The fundamental operation of eden-det (A, φ) is cleary nh-elim(A, H). From the pseudo code for nh-elim$(A \in H)$ (Algorithm 1), it is also clear that the number of operations executed by nh-elim(A, H) is a function of the number of cells $n = |V|$. We will show that this operation is in $\Theta(n)$.

The four lines within the innermost loop of nh-elim$(A \in H)$ are only dependent on the number of neighborhood configurations. Without loss of generality, we assume $\forall i |h_i| = k$, thus the construction of Θ_i and Θ_j require searching only a single column of H. That is, $C_\Theta = c_0|\Sigma^k|$ where $c_0 \approx k$ is the number of operations to evaluate $\theta_i^{|i \equiv j|}$. The construction of ζ_i is dependent only on the size of the Θ's, hence $C_\zeta \leq C_\Theta$. Furthermore, the number of elements in H is equal the number of elements in $\zeta_i \leq |\Sigma^k|$. Thus the total operation count within the inner loop is given by,

$$C_{inner} = 2C_\Theta + C_\zeta + |\zeta| \approx 3|\Sigma^k| \qquad (3)$$

Given Equation (3) we can derive the total operation count for nh-elim$(A \in H)$.

$$\begin{aligned} C_{NH}(n) &= \sum_{i=1}^{n} \sum_{j=1}^{k} C_{inner} \qquad (4) \\ &= kC_{inner}n \\ &\approx 3k|\Sigma^k|n \end{aligned}$$

Therefore $C_{NH}(n) \in \Theta(n)$.

4.2 Best case

We now consider the best case time complexity of eden-det (A, φ). The best case occurs when there exist few possible $i \square j$-consistent pairs for some sub-sequence in φ. This is very common in cellular automata in which Langton's λ [6] is small. An example of this is when A is the elementary cellular automaton rule 2, and φ has a contiguous sequence of 1's.

In this special case, only ph1$(A \square H)$ (Algorithm 3) will be required. Furthermore a column of zeros will developed very quickly as each iteration will eliminate at least one possible configuration from the unnatural area (due to few or no $i \square j$-consistent neighborhood pairs), that is $I < |\Sigma^k|$ where I is the number of iterations of the while loop. Using the results from Equation (4) we have,

$$\begin{aligned} C_{best}(n) &= \sum_{i=1}^{|\Sigma^k|} C_{NH}(n) \\ &= |\Sigma^k| C_{NH}(n) \\ &\approx 3k|\Sigma^k|^2 n \end{aligned} \tag{5}$$

Therefore $C_{best} \in \Omega(n)$.

4.3 Worst case

For the worst case we must consider the full expression for the number of operations executed by eden-det (A, φ). This is given by,

$$C_{ops}(n) = \sum_{t=1}^{J} \left[\underbrace{\sum_{j=1}^{I} C_{NH}(n)}_{\text{ph1}(A \square H)} + \underbrace{\sum_{j=1}^{|V|} \sum_{i=1}^{|\Sigma^k|} C_{NH}(n)}_{\text{ph2}(A \square H)} \right] \tag{6}$$

where J and I simply denote the number of iterations taken by the conditional loops. We require an upper bound on these loops.

In Section 3.3 we noted that the number of 0's in H can never decrease. Now we also note that if the number of 0's in H does not increase after an execution of ph2$(A \square H)$ (Algorithm 4) then eden-det (A, φ) terminates with GoE = false. Hence for the worst case we must assume that the number of 0's decreases by exactly one. Furthermore, every iteration of ph1$(A \square H)$ will either increase the number of 0's, terminate eden-det (A, φ) with GoE = true, or continue to an iteration of ph2$(A \square H)$. Since $H \in \{0 \square 1\}^{|\Sigma^k| \times n}$, it must hold that

$$J(I + 1) \leq |\Sigma^k| n \square \tag{7}$$

We want to maximize the value of J as it has the greater effect on the total number calls to nh-elim(A,H). If we assume the upper bound is reachable, then as $I \to 1$ we have $J \to \frac{|\Sigma^k|}{2}n$. We can apply this result to Equation (6) to obtain an upper bound on $C_{ops}(n)$,

$$\begin{aligned} C_{ops}(n) &\leq \sum_{t=1}^{\frac{|\Sigma^k|}{2}n} \left[C_{NH}(n) + \sum_{j=1}^{|V|} \sum_{i=1}^{|\Sigma^k|} C_{NH}(n) \right] \\ &= \frac{|\Sigma^k|}{2}n \left[C_{NH}(n) + |V||\Sigma^k| C_{NH}(n) \right] \\ &= \frac{|\Sigma^k|}{2} \left[|\Sigma^k| n^2 + n \right] C_{NH}(n) \end{aligned}$$

Furthermore, we have already shown that $C_{NH} \in \Theta(n)$. Therefore $C_{worst} \in O(n^3)$.

4.4 Average case

Best and worst case bounds are important but of limited practical use without an indication of the likelihood of eden(A,φ) instances which cause these bounds to occur. In this section we will show, using empirical data, that the average case is quadratic in time.

Consider Equation (6), the values affecting the computational complexity are the number of iterations taken by the guard loops and whether ph2(A,H) needs to be executed. As in Section 4.3, we will denote the number of outer loops as J and the number of ph1(A,H) loops as I. Furthermore, we denote the number of iterations in which ph2(A,H) is executed as K.

We took random eden(A,φ) instances for $|V| = n = 2^i$, $2 \leq i \leq 13$, where G is a single circuit. For each value of n over 1000 samples were taken. The values of I,J, and K were counted for each sample. The expected values computed from these samples are shown in Figure 4

From Figure 4 we can derive the overall expected values E(I) = 2.88, E(J) = 1.06, and E(K) = 0.25. So it is reasonable to approximate the average case as follows,

$$\begin{aligned} C_{average}(n) &\approx \left(\sum_{i=1}^{3} C_{NH}(n) \right) \times \Pr(\neg K) \\ &\quad + \left[\sum_{i=1}^{3} C_{NH}(n) + \sum_{j=1}^{|V|} \sum_{i=1}^{|\Sigma^k|} C_{NH}(n) \right] \times \Pr(K) \\ &= (3C_{NH}(n)) \times \frac{3}{4} + \left[3C_{NH}(n) + |\Sigma^k| n C_{NH}(n) \right] \times \frac{1}{4} \\ &= \frac{|\Sigma^k|}{4} n C_{NH}(n) + 3C_{NH}(n) \end{aligned}$$

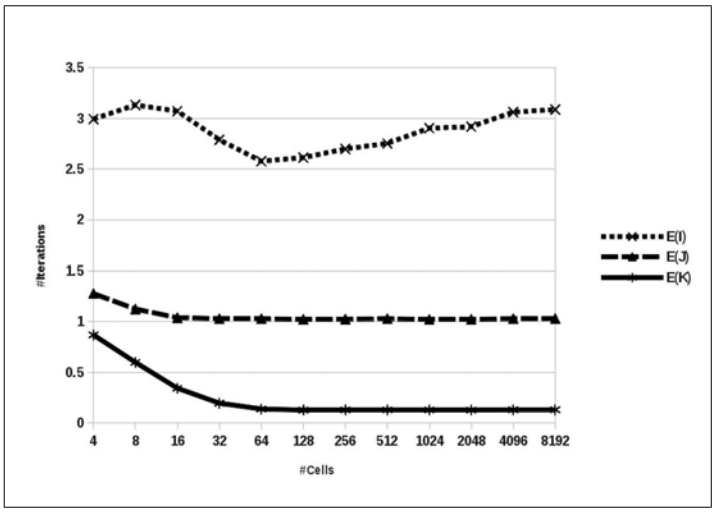

Figure 4. Expected Iteration Values. Based on 1000 random samples for each number of cells.

Since $C_{NH}(n) \in \Theta(n)$, the approximate overall expected time complexity is $C_{average}(n) \in \Theta(n^2)$. Section 4.5 provides further experimentation to validate this approximation.

4.5 Experimental results

For validation of the average case we took a new random sample of 1000 instances of eden(A,φ) for $|V| = n = 2^i \square 2 \leq i \leq 13$. For each sample the average runtime of 5 separate runs was taken. Results were separated into two groups based on whether eden-det (A,φ) returned with GoE = true or GoE = false. The resulting average run times are shown in Figure 5.

Note that on average the runtime when GoE = false is approximately 16 times the runtime when GoE = true. This is because only a Garden-of-Eden configuration φ_e can cause eden-det (A,φ_e) to return before Phase 2 is executed, which will complete in $O(n)$ operations.

To confirm that the curves in Figure 5 are in fact quadratic, we can take the ratio $R = \frac{C(2^{i+1})}{C(2^i)}$ where $C(n)$ is the average runtime as a function of the number of cells n. We would expect $R \approx 4$ for a quadratic (i.e., doubling the input takes 4 times longer). Figure 6 shows the R for there samples taken for Figure 5.

From Figure 6 it is clear that $R \approx 4$ (Considering that $R \approx 2$ for linear and $R \approx 8$ for cubic). This provides support for our approximate average time complexity for eden-det (A,φ) that we provided in

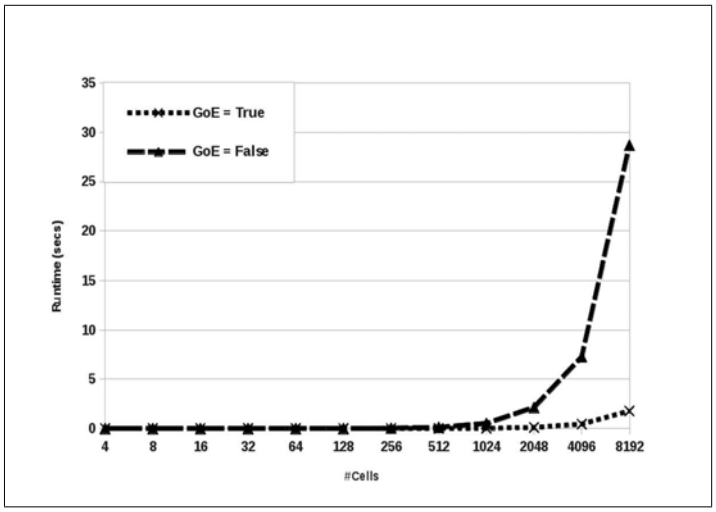

Figure 5. Run times for eden-det (Algorithm 5).

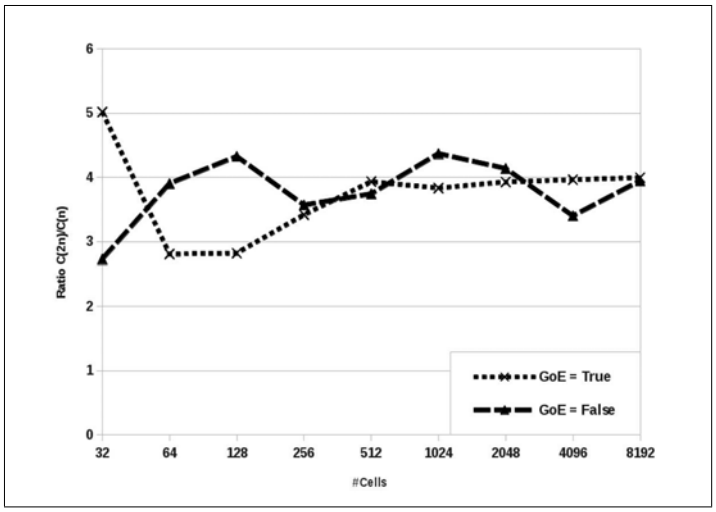

Figure 6. Ratio of run times C(2n)/C(n)).

Section 4.4.

5. Comparison with Wuensche and Lesser's reverse algorithm

In this section, we will compare the performance of eden-det against the reverse algorithm developed by Wuensche and Lesser [20]. For the sake of simplicity, we will restrict the comparison to the simplest form of cellular automata; that of finite elementary cellular automata with periodic boundary condition. As a result we must emphasize that the following discussion and analysis relates specifically to Wuensche and Lesser's one dimensional reverse algorithm [20] and not Wuensche's more general reverse algorithm which applies to random boolean networks and graph cellular automata [19, 17]. The results of this analysis, however, can certainly be generalized to the graph cellular automata case.

For a finite elementary cellular automata with periodic boundary conditions A, a configuration φ^t and a partial pre-image φ^{t-1} in which the first i cell states are known, Wuensche and Lesser's method is described as follows [20]:

1. If $g(\varphi^{t-1}_{i-1}\varphi^{t-1}_i 0) = g(\varphi^t_{i-1} - 1\varphi^t_i 1) \neq \varphi^t_i$, then abandon the partial pre-image. Resume derivation of next partial pre-image (go to step 5).
2. If $g(\varphi^{t-1}_{i-1}\varphi^{t-1}_i 0) \neq g(\varphi^t_{i-1} - 1\varphi^t_i 1)$, then φ^{t-1}_{i+1} can be uniquely determined. Proceed with next cell (go to step 1).
3. If $g(\varphi^{t-1}_{i-1}\varphi^{t-1}_i 0) = g(\varphi^t_{i-1} - 1\varphi^t_i 1) = \varphi^t_i$, then φ^{t-1}_{i+1} could be 0 or 1. Push the partial pre-image $(\varphi^{t-1}_0\varphi^{t-1}_1 \cdots \varphi^{t-1}_i 1)$ onto the pre-image queue to be processed later and continue with $\varphi^{t-1}_{i+1} = 0$.
4. When $i = n - 1$ check that $g(\varphi^{t-1}_{n-2}\varphi^{t-1}_{n-1}\varphi^{t-1}_0) \neq g(\varphi^{t-1}_{n-1}\varphi^{t-1}_0\varphi^{t-1}_1)$ then abandon this pre-image, otherwise add to the valid pre-image list.
5. Take a new partial pre-image from the queue and continue processing (step 1).
6. When the partial pre-image queue is empty, all possible pre-images starting with the start values of $\varphi^{t-1}_0\varphi^{t-1}_1$ are derived. Repeat for all possible $\varphi^{t-1}_0\varphi^{t-1}_1$.

Note that the primary purpose of Wuensche and Lesser's method is the construction of all valid pre-images, but it can be utilised directly to compute the solution to the Eden problem. Clearly, we can assert GoE = false as soon as a valid pre-image is found. We need not compute all of them. GoE = true will be asserted when not pre-images are found.

5.1 Worst case complexity analysis

With a brief description of Wuensche and Lesser's one dimensional reverse algorithm, we can now show that the worst case computation time is not bounded by a polynomial in the number of cells n. Consider Algorithm 6 which depicts Wuensche and Lesser's method modified for solving the Eden problem without computing all pre-images.

Algorithm 6 `reverse(A,φ)`: Wuensche and Lesser's Reverse Algorithm.

```
GoE ← true
for all (p1,p2) ∈ {(0,0),(0,1),(1,0),(1,1)} do
  φ^{t-1} ← (p1,p2)
  Q ← {φ^{t-1}}
  while Q ≠ {} do
    φ^{t-1} ← pop(Q)
    x = |φ^{t-1}| - 1
    for all i ∈ [x,n] do
      T0 ← g(φ_{i-1}^{t-1},φ_i^{t-1},0)
      T1 ← g(φ_{i-1}^{t-1},φ_i^{t-1},1)
      if T0 = T1 ≠ φ_i^t then
        break for loop
      else
        if T0 ≠ T1 then
          if T0 = φ_i^{t-1} then
            φ^{t-1} ← φ^{t-1} ∪ {0}
          else
            φ^{t-1} ← φ^{t-1} ∪ {1}
          end if
        else
          push(Q,φ^{t-1} ∪ {1})
          φ^{t-1} ← φ^{t-1} ∪ {0}
        end if
      end if
    end for
    Tn ← g(φ_{n-1}^{t-1},φ_n^{t-1},φ_1^{t-1})
    T1 ← g(φ_n^{t-1},φ_1^{t-1},φ_2^{t-1})
    if Tn = T1 then
      GoE ← false
      return GoE
    end if
  end while
end for
return GoE
```

Let C_{inner} denote the number of operations performed on a single iteration of the innermost loop in `reverse(A,φ)`. Without loss of

generality, we will assume C_{inner} is a constant[4].

Now, let $C_x(n)$ denote the number of operations required to complete ignoring partial pre-images already in Q before reaching the x-th cell in the current pre-image. We can express $C_x(n)$ as the following recurrence relation,

$$C_x(n) = \sum_{i=x}^{n} C_{inner} + e(x) \sum_{i=x+1}^{n} a(i) C_i(n)$$

where $e(x) = 0$ if $T_0 = T_1 \Leftrightarrow \varphi_x^{t-i}$ otherwise $e(x) = 1$, and $a(x) = 1$ if $T_0 = T_1 = \varphi_x^{t-1}$ otherwise $a(x) = 0$.

The worst case for $C_x(n)$ occurs when the number of partial pre-images being pushed onto the queue is every iteration. In this case, we have $\forall i > x\,(a(i) = 1 \wedge e(i) = 1)$ and the recurrence relation becomes,

$$C_x(n) = \sum_{i=x}^{n} C_{inner} + \sum_{i=x+1}^{n} C_i(n)\,.$$

We can now solve this recurrence relation. First consider expanding the $C_{x+1}(n)$ term in the summation,

$$\begin{aligned}
C_x(n) &= \sum_{i=x}^{n} C_{inner} + \sum_{i=x+1}^{n} C_i(n) \\
&= \sum_{i=x}^{n} C_{inner} + C_{x+1}(n) + \sum_{i=x+2}^{n} C_i(n) \\
&= C_{inner} + \left(\sum_{i=x+1}^{n} C_{inner} + \sum_{i=x+1}^{n} C_{inner} + \sum_{i=x+2}^{n} C_i(n) \right) \\
&\quad + \sum_{i=x+2}^{n} C_i(n) \\
&= \sum_{i=x}^{n} C_{inner} + \left(\sum_{i=x+1}^{n} C_{inner} + \sum_{i=x+2}^{n} C_i(n) \right) + \sum_{i=x+2}^{n} C_i(n) \\
&= C_{inner} + 2 \sum_{i=x+1}^{n} C_{inner} + 2 \sum_{i=x+2}^{n} C_i(n)\,.
\end{aligned}$$

[4] Clearly this is not true in reality, but instead $3 \leq C_{inner} \leq 6$

Now, expending the $C_{x+2}(n)$ term,

$$\begin{aligned}
C_x(n) &= C_{inner} + 2\sum_{i=x+1}^{n} C_{inner} + 2\sum_{i=x+2}^{n} C_i(n) \\
&= C_{inner} + 2\sum_{i=x+1}^{n} C_{inner} + 2C_{x+2}(n) + 2\sum_{i=x+3}^{n} C_i(n) \\
&= C_{inner} + 2C_{inner} + 2\sum_{i=x+2}^{n} C_{inner} \\
&\quad + 2\left(\sum_{i=x+2}^{n} C_{inner} + \sum_{i=x+3}^{n} C_i(n)\right) + 2\sum_{i=x+3}^{n} C_i(n) \\
&= C_{inner} + 2C_{inner} + 4\sum_{i=x+2}^{n} C_{inner} + 4\sum_{i=x+3}^{n} C_i(n).
\end{aligned}$$

Repeating this process yields,

$$\begin{aligned}
C_x(n) &= C_{inner} + 2C_{inner} + 4C_{inner} + \cdots + 2^{n-x}C_{inner} \\
&= C_{inner}\sum_{i=x}^{n} 2^{i-x}.
\end{aligned}$$

The worst case for $reverse(A,\varphi)$ requires that $C_2(n)$ operations be executed four times,

$$\begin{aligned}
C_{worst} &= 4C_2(n) \\
&= 4C_{inner}\sum_{i=2}^{n} 2^{i-2} \\
&= C_{inner}\sum_{i=2}^{n} 2^{i}
\end{aligned}$$

hence $reverse(A,\varphi)$ is in $O(2^n)$. It is worth noting that this worst case can only be achieved if the φ is a Garden-of-Eden configuration, and the cell which determines this is the n-th cell. For example, $\varphi = (0,0,\ldots,0,1,1)$ for the elementary cellular automaton rule 2. However, according to Wuensche and Lesser [20] the average case is orders of magnitude better. We confirm this experimentally in Section 5.2.

5.2 Experimental comparison

We benchmarked $eden\text{-}det(A,\varphi)$ against $reverse(A,\varphi)$. Each experiment consisted of solving the Eden problem for 1000 random configurations. Experiments were performed for both $eden\text{-}det(A,\varphi)$ and

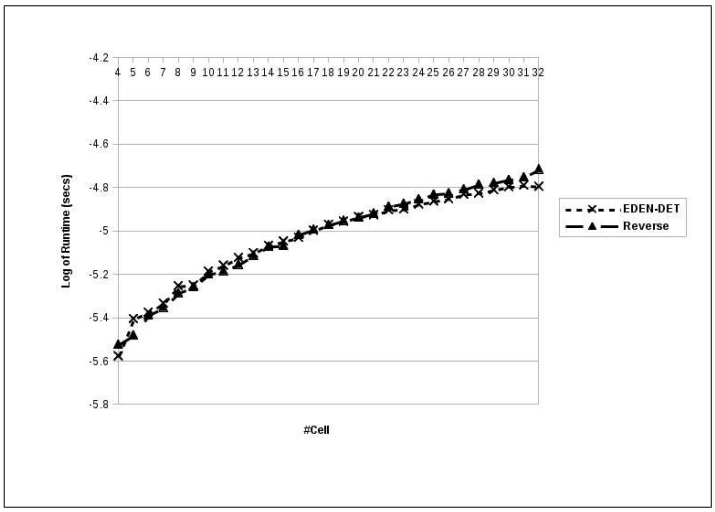

Figure 7. Comparison of eden-det (A,φ) against reverse(A,φ) using a random sampling of configurations.

reverse(A,φ) using all the elementary cellular automata with cell counts ranging from 4 to 32. As shown in Figure 7, the benchmark average case is effectively the same order of magnitude for both methods.

The worst case for reverse(A,φ) is only approached for Garden-of-Eden configurations which is nearly identical to a non Garden-of-Eden configuration only differing in the last few cells. This is more likely to be possible with sparse configurations (i.e., very few 1 states compared with 0 states). If we restrict the random sample of test configurations to that of sparse configurations, then the probability of selecting a configuration which degrades the performance of reverse(A,φ).

Figure 8 indicates that the benchmark results are very different when we restrict the configuration sample this way. Such cases place a limitation on the usability of reverse(A,φ) for large cell counts[5]. The performance of eden-det (A,φ), however, is hardly affected by such sparse configurations.

The main difference in our approach which provides such a large improvement in the worst case performance is the neighborhood elimination step. This operation performance is not affected by shifts (or rotations in higher dimension) in the same configuration, because it treats each cell neighborhood independently of each other. As a result, eden-det (A,φ) provides a solution to the Eden problem which is scalable to very large cellular automata. eden-det (A,φ) could be considered as a more stable alternative to Wuesnche and Lesser's re-

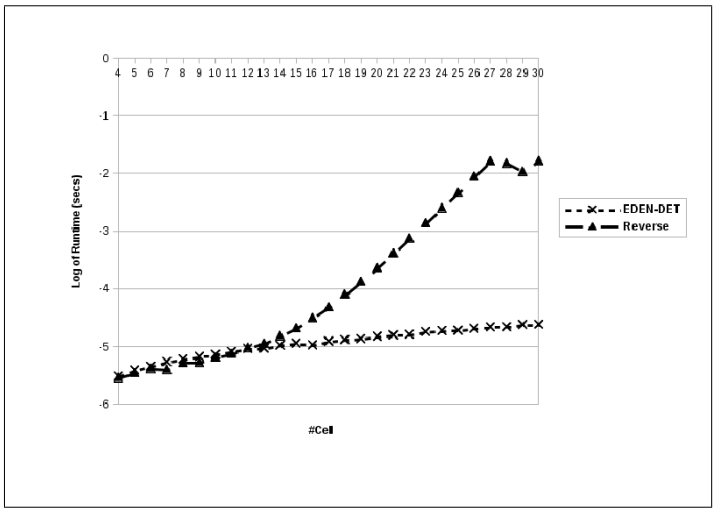

Figure 8. Comparison of eden-det (A,φ) against reverse(A,φ) using a random sampling of sparse configurations.

verse(A,φ) as the worst case is vastly improved without degrading the average case.

6. Algorithm correctness

In this section, we discuss the correctness of the eden-det (A,φ) in solving the Eden problem for graph cellular automata. We are able to show that eden-det (A,φ) is completely correct for graphs with a single cycle. For graphs with more than one cycle it is possible for incorrect results to be returned[6] (i.e., false negatives), however we show that these cases are rare.

It is first worth discussing the correctness of the solution when eden-det (A,φ) returns with GoE = true. This result will never occur if φ has a pre-image (i.e., false positives cannot occur). This is because elements in H are only ever set to 0 when there is no $i\,\square\,j$-consistent pair in a neighbor cell. If GoE = true is returned then at some point there must have existed an i such that $\forall j\ H_{j\,\square\,i} = 0$ (i.e., a cell has no possible $i\,\square\,j$-consistent neighborhood configurations). For φ to have a pre-image each cell must have at least one $i\,\square\,j$-consistent neighborhood

[5] As the cell count increases any configuration with a relatively small sparse sub-sequence could render the Eden problem computationally intractable for reverse$(\mathbb{A},\varphi)$

[6] If this were not so the title of this paper would be "P = NP"!

configuration. Therefore only a true Garden-of-Eden configuration can cause GoE = true to be returned.

When eden-det (A,φ) returns with GoE = false there are possible false identifications. That is, it is possible for a Garden-of-Eden configuration to cause GoE = false to be returned. However, this rare case is only a possibility when the graph G has more than one cycle.

We will now show that eden-det (A,φ) returning GoE = false is always correct if G contains only one cycle. Consider H which has reached a non-zero steady state after executing ph1(A,H). If we assume a neighborhood configuration $H_{j,i} = 1$ (see Section 3.3), and carry out an iteration of nh-elim(A,H) we are essentially propagating the assumption around the cycle of G. When this propagation returns to i then there are only two possibilities: 1) The assumed $H_{j,i}$ is not eliminated meaning a chain of i–j-consistent pairs can be constructed (i.e., a pre-image can exist under this assumption), and 2) The assumed $H_{j,i}$ is eliminated, hence ψ_j cannot contribute to any pre-image. Since eden-det (A,φ) only returns GoE = false when every element in H has passed assumption testing, we can conclude this can only occur if φ does in fact have a possible pre-image. Therefore eden-det (A,φ) is completely correct for G with a single cycle.

These correctness results for the single cycle (i.e., 1-d) case have also been supported by experimental results. We executed eden-det (A,φ) on the entire configuration space for all elementary cellular automata where n = [4,8,16]. Each return value was validated via a brute force search for a pre-image. This resulted in a 100% success rate.

Unfortunately, things are not so easy for G with multiple cycles. The assumption testing method we apply in ph2(A,H) is really only powerful enough to test consistency within a single cycle. It may be possible for every $H_{j,i}$ to pass the assumption test but any choice made from one cycle breaks consistency in another. Hence a complete solution would require looking at pairs of cycles, triples of cycles etc.[7]. This is likely the result of the NP-complete nature of the Eden problem in more than one dimension.

Again, we look to empirical data to show that in the majority of cases the single cycle accuracy is all we need. This time over 170,000 random instances of eden(A,φ) (for a fixed choice of rules representing Wolfram Classes i,ii, and iii [15]) were taken as inputs[8]. Every result was compared to a brute force approach.

We found that for Class i cellular automata (i.e., point attractors) no false negatives ever seem to occur. Rules that fall under Class ii

[7] Of course we do not have a rigorous proof of this. If we did, the title of this paper would be "P = NP"!

[8] The topology of the graph G was equivalent to a dodecahedron. Since |V| = 20, the complete configuration space is 2^{20}.

(i.e., simple structures, maybe periodic) had a low number of false negatives; around 0.01%. Class iii cellular automata (i.e., chaotic) are a different story, with around 16% of cases in which eden-det (A,φ) returned GoE = false were incorrect[9]. Over all samples, the false negative rate was around 10%.

False negatives can be detected without resorting to a brute force sweep. As previously stated in Section 3.3, the final state of H completely encodes all possible pre-images. Neighborhoods in H can be stitched together using a method similar to Wuensche's general reverse algorithm [19], if no pre-image can be constructed then we have detected a false negative. In light of this, our algorithm could also be considered as a search reduction step to be used prior to invoking Wuensche's general method. Combined, this would provide a completely correct and more efficient method for constructing configuration transition graphs.

7. Conclusion

In this paper we have presented an efficient algorithm (i.e., average case in $\Theta(n^2)$), eden-det (A,H), for solving the Eden problem for graph cellular automata. By changing the topology of the graph G, the Eden problem can be solved for all classes of deterministic discrete dynamical systems (e.g., regular cellular automata, and random boolean networks). This analysis provides a firm foundation for further study of the global dynamics of discrete dynamical systems.

Appendix

A. Proof of Theorem 1

Proof. First consider the equality,

$$\theta_i^{\,i\Xi j}(\psi_n) = \theta_j^{\,i\Xi j}(\psi_m).$$

Given Definition 9, we can expand the above expression. This yields,

$$\equiv \forall s.(\psi'_{n,s} = \psi'_{m,s} \wedge \exists p.(\psi_{n,p} = \psi'_{n,s} \wedge y = h_{i,p} \wedge y = {}^{i}\Xi^{j}_{s}) \wedge \exists q.(\psi_{m,q} = \psi'_{m,s} \wedge z = h_{j,q} \wedge z = {}^{i}\Xi^{j}_{s})).$$

[9] It is interesting to note that it is only Class iii cellular automata that seem to cause ph2(A,H) to be executed in the 1-d case.

This can be reduced using predicate calculus,

$$\begin{aligned}
&\equiv \forall s(\psi'_{n,s} = \psi'_{m,s} \wedge \exists p(\psi_{n,p} = \psi'_{n,s} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s}) \\
&\quad \wedge \exists q(\psi_{m,q} = \psi'_{m,s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s})) \\
&\equiv \forall s(\psi'_{n,s} = \psi'_{m,s} \wedge \\
&\quad \exists p,q(\psi_{n,p} = \psi'_{n,s} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge \psi_{m,q} = \psi'_{m,s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s})) \\
&\equiv \forall s(\exists p,q(\psi'_{n,s} = \psi'_{m,s} \wedge \psi_{n,p} = \psi'_{n,s} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \\
&\quad \wedge \psi_{m,q} = \psi'_{m,s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s})) \\
&\equiv \forall s(\exists p,q(\psi_{n,p} = \psi_{m,q} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s}))
\end{aligned}$$

Now let $C^0(h_i) = \psi_n$ and $C^0(h_j) = \psi_m$, hence $C^0(h_{i,p}) = \psi_{n,p}$ and $C^0(h_{j,q}) = \psi_{m,q}$

$$\begin{aligned}
&\models \forall s(\exists p,q(\psi_{n,p} = \psi_{m,q} \wedge h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s}) \wedge C^0(h_{i,p}) = \psi_{n,p} \\
&\quad \wedge C^0(h_{j,q}) = \psi_{m,q}) \\
&\equiv \forall s(\exists p,q(h_{i,p} = {}^{i}\Xi^{j}_{s} \wedge h_{j,q} = {}^{i}\Xi^{j}_{s}) \wedge C^0(h_{i,p}) = C^0(h_{j,q}))
\end{aligned}$$

This satisfies our definition of i,j-consistency (i.e., Definition 8). Therefore ψ_n and ψ_m are i,j-consistent. ■

References

[1] S. Bandini, G. Mauri, and R. Serra, "Cellular automata: From a theoretical parallel computational model to its application to complex systems," Parallel Computing, 27 (2001) 539–553.

[2] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Streans, and M. Thakur, "Predecessor existence problems for finite discrete dynamical systems," Theoretical Computer Science, 386 (2007) 3–37.

[3] M. Cook, "Universality in elementary cellular automata," Complex Systems, 15 (2004) 1–40.

[4] N. Fates, "Critical phenomena in cellular automata: Perturbing the update, the transitions, the topology," Acta Physica Polonica B, Proceedings Supplement, 3 (2010) 315–325.

[5] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," Journal of Theoretical Biology, 22 (1969) 439–467.

[6] C. G. Langton, "Computation at the edge of chaos: phase transitions and emergent computation," Physica D, 42 (1990) 12–37.

[7] C. Marr, and M. T. Hütt, "Outer-totalistic cellular automata on graphs," Physics Letters A, 373 (2009) 546–549.

[8] C. Marr, and M. T. Hütt, "Topology regulates pattern formation capacity of binary cellular automata on graphs," Physica A, 354 (2005) 641–662.

[9] S. Seif, "Constrained Eden," Complex Systems, 18 (2009) 379–385.

[10] K. Sutner, "On the computational complexity of finite cellular automata," Journal of Computer and System Science, 50 (1995) 87–97.

[11] M. Tomassini, "Generalized automata networks," Lecture Notes in Computer Science, 4173 (2006) 14–28.

[12] D. J. Warne, `GCALab` [C program for analysis of graph cellular automata] (available on GitHub, https://github.com/davidwarne/GCALab).

[13] S. Wolfram, A new kind of science (Wolfram Media Inc., Champaign, IL, 2002).

[14] S. Wolfram, "Complex Systems Theory," Emerging Syntheses in Science: Proceedings of the Founding Workshops of the Santa Fe Institute (Addison-Wesley, 1988)

[15] S. Wolfram, "Universality and complexity in cellular automata," Physica D, (1984) 1–35.

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

[17] A. Wuensche, Exploring discrete dynamics (Luniver Press, Frome, UK, 2011).

[18] A. Wuensche, "Genomic regulation modeled as a network with basins of attraction," Pacific Symposium on Biocomputing'98, Singapore (1998) 89–102.

[19] A. Wuensche, "The ghost in the machine: basins of attraction of random boolean networks," in Artificial Life III, edited C.G. Langton (Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, Reading, MA, 1994).

[20] A. Wuensche, and M. Lesser, The global dynamics of cellular automata (Addison-Wesley, Reading, MA, 1992).

A. PUBLICATION: AN EFFICIENT ALGORITHM FOR THE DETECTION OF EDEN

Appendix B

The Graph Cellular Automata Laboratory

The majority of the computational tasks in this project were performed using the Graph Cellular Automata Laboratory (GCALab) which was custom built for this project. This Appendix provides a summary of this software.

B.1 Overview

The GCALab software is an interactive computational tool for the creation, simulation, analysis, and visualisation of graph cellular automata. GCALab is designed to perform many graph cellular automata calculations in parallel while maintaining an interactive command prompt to the user. This allows the user to send a calculation to a back-end compute thread, and then continue to analyse/visualise the results of another calculation.

GCALab is a free Open Source project and is licensed under the GNU Public License Version 3. The latest version of the source code may be access via GitHub at https://github.com/davidwarne/GCALab.git. The software is still very much under development will new functionality expected in the future.

B.2 Architecture

GCALab is based on a flexible modular Architecture. Every user interface command, and back-end compute operation is abstracted to a consistent function callback interface. Due to design flexibility, new functionality may be easily implemented and integrated into the whole system. GCALab has also been built to take advantage of parallelism available to almost all modern day processors. The interactive font-end acts like a "master" forwarding operations to back-end compute threads (attached to a work-space). This allows the user to continue to work and experiment while waiting for results to complete.The modular and parallel architecture of GCALab is shown in Figure B.1.

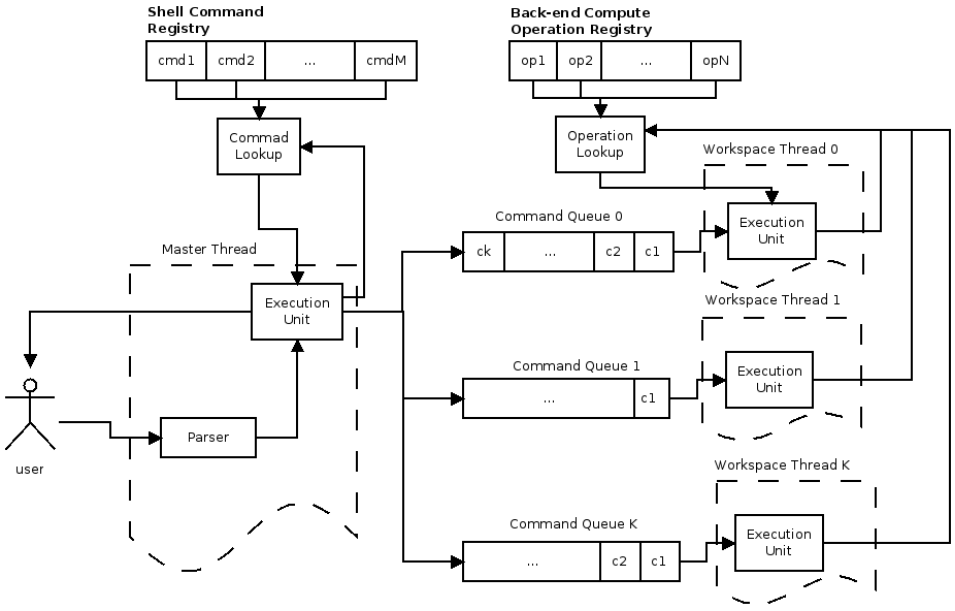

Figure B.1: Architecture of GCALab

From Figure B.1 it can be seen that the interactive shell runs as a "master" thread. This simply parses user commands from the command line and forwards

the parse string to an execution unit. If the parsed string represents a back-end compute operation then the execution unit pushes the operation (and input arguments) to the command queue of the target workspace. Each workspace has its own compute thread which will continuously poll the command queue for an assigned task,when the task is complete it sends a signal back to "master" to inform the user. Currently each workspace is designed to be a logically separate compute module with does not communicate with other workspaces, this enables linear speed-up to occur if one workspace is assigned per available CPU core.

In Figure B.1, it can been seen that all shell commands and back-end compute operations are provided as callbacks. This is how GCALab's extensible and flexible architecture is realised.

An interactive shell command implements the following interface,

```
char (*f)(int argc, char** argv);
```

where `argc` is the number of command line arguments, and `argv` is a pointer to the string arguments as parsed directly from the command prompt, the function should also return the macro `GCALab_SUCCESS` when the command has been completed successfully, otherwise an appropriate error macro may be used. New interactive shell commands may be created and added to GCALab using the `GCALab_Register_Command()` function.

A back-end compute operation must implement the following interface.

```
char (*f)(unsigned char ws_id, unsigned int trgt_id,
          int nparams, char** params, GCALabOutput **res);
```

where `ws_id` is a reference to the workspace that the command was queued to, `trgt_id` is the reference to a cellular automata or result structure, `nparams` is the number of command line arguments (excluding command name and target id) supplied by the user, `params` is a pointer to the string arguments parsed from the command prompt, and `res` is a pointer to a pointer of a results structure that must be created in the function call. Just as with shell commands `GCALab_SUCCESS` should be returned on a valid completion and an appropriate error macro returned otherwise.

Currently GCALab has only been tested on Linux/Unix based systems. However, the only dependencies that GCALab requires is POSIX threads, OpenGL (any version >= 1:1), and FreeGlut[1].

B.3 Modes of Execution

A user may start up GCALab in one of three modes. They are,

- **Interactive Mode** This is the standard mode of operation for GCALab. This is a text command prompt style interactive "shell" to GCALab functions.

- **Batch Mode** In this mode the user can supply a "script" file containing a list of commands as the user would enter them in the interactive mode, in batch mode GCALab will compute these in order "offline". This is especially useful when spanning many calculations across a resource such as a compute cluster.

- **Graphics Mode** This mode runs an OpenGL Cellular Automata visualisation window. The user can interacted with this viewer in 3D and in supports animation an interactive modification of cellular automata configurations. The user can also drop to a standard interactive shell for more advanced commands.

B.4 Commands

GCALab is primarily a commandline based application. It implements three kinds of commands: interactive shell commands, back-end compute operations, and graphics mode commands. This Section lists these commands with a brief

[1] GCALab can be compiled without graphics support, in which case only POSIX threads is required

summary, the reader is referred to the GCALab documentation for further details[1].

B.4.1 Interactive Shell Commands

Interactive shell commands manage the creation of workspaces, file IO, pushing compute operations to workspace queues, and printing of results.

- `new-work` - Creates a new workspace with a given number of cellular automata slots.
- `print-work` - Print the current workspace.
- `list-work` - Print summary information of all workspaces.
- `ch-work` - Changes current workspace.
- `q-cmd` - Enqueues a back-end compute operation to the current command queue.
- `del-cmd` - Sets a given queued back-end compute operation to NOP and hence will be be ignored.
- `exec-q` - The current queue will start processing.
- `stop-q` - The current queue will pause after completion of the current operation.
- `print-ca` - Prints a given graph cellular automaton in the current workspace.
- `print-st` - Prints a graph cellular automaton evolution as a space-time pattern.
- `print-res` - Prints a given result data set.
- `quit` - Exits GCALab.
- `help` - Prints the help menu.

[1] Available on GitHub at https://github.com/davidwarne/GCALab.git.

- `list-cmds` - Prints available back-end compute compute operations.

B.4.2 Back-end Compute Operations

Back-end compute command execute a calculation on a given cellular automaton and return the result back to workspace memory, back-end compute operations always execute on compute threads asynchronously with the user interface (shell) thread.

- `nop` - No Operation.
- `load` - Loads a graph cellular automaton from file into the current workspace.
- `save` - Save a graph cellular automaton or result data to file
- `sim` - Simulates the given graph cellular automaton to a given time-step.
- `gca` - Creates a new graph cellular automaton in the current workspace.
- `entropy` - Computes entropy measures of the given graph cellular automaton.
- `param` - Computes complexity parameters such as Langton's lambda.
- `pre` - Computes pre-images of the current configuration of the given graph cellular automaton.
- `freq` - Computes state frequency histogram for each cell in the given graph cellular automaton.
- `pop` - Computes the non-quiescent population density over time.

B.4.3 Graphics Mode Commands

We running GCALab in graphics mode and extra set of single keystroke commands and mouse controls enable to user to interact with a 3D visualisation of the selected graph cellular automaton.

B.4.3.1 Keyboard Commands

- `c` - Drop to the GCALab Shell.
- `>` - Switch viewer to next workspace.
- `<` - Switch viewer to previous workspace.
- `]` - Switch viewer to next graph cellular automaton (in the current workspace).
- `[` - Switch viewer to the next graph cellular automaton (int the current workspace).
- `Q` - Exits GCALab.
- `s` - Steps the viewed graph cellular automaton forward in time.
- `l` - Toggles colour mode between state-base and neighbourhood-based.
- `p` - Toggles configuration edit mode.
- `m` - Toggles mesh visibility.

B.4.3.2 Mouse Controls

- **Left-button** Enables mouse to rotate the scene, or sets the selected cell to a non-quiescent state when in edit mode.
- **Middle-button** Enables mouse to zoom the scene.
- **Right-button** Enables mouse to translate the scene, or sets the selected cell to a quiescent state when in edit mode.

B.5 Usage Example

This section presents a simple example of using the GCALab interactive shell. The particular use case scenario is computing a rough approximation of the Shannon entropy for elementary cellular automaton *rule 110*.

1. When executing GCALab in interactive mode the user is presented with the interactive shell prompt.

```
+++ Welcome to Graph Cellular Automata Lab! +++

The Graph Cellular Automata Lab is a multi-threaded,
flexible, analysis tool for the exploration of cellular automata
defined on graphs.

Version:   0.19
Author:    David J. Warne
School:    School of Electrical Engineering and Computer Science,
           The Queensland University of Technology, Brisbane, Australi
Contact:   david.warne@qut.edu.au
url:       https://github.com/davidwarne/GCALab.git

GCALab v 0.19  Copyright (C) 2013  David J. Warne
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions.

Type 'help' to get started.
-->
```

2. First create a new workspace with enough memory for up to 10 graph cellular automata.

```
-->new-work 10
New Workspace created! ID = 0
```

3. Next use the gca command to create an instance of *rule 110* with 32 cells[1].

```
-->gca 0 -s 2 -eca 32 3 110
```

4. Now compute the Shannon entropy based on 100 samples each with 10 time-steps.

```
-->entropy 0 -n 100 -t 10 -e Shannon
```

5. And finally print the results.

```
-->print-res 0
type: 0
id: (0):S
data length: 1
0.867771
```

This example is designed to be a "Hello World" introduction to using GCALab; it is by no means representative of all that is possible with the software or even the commands used. For full details on all the commands refer to the GCALab documentation.

[1] Note that gca is actually a back-end compute operation, but we did not use the q-cmd command. This is because the q-cmd is optional as the GCALab interpreter can tell the difference, but a user may use it for clarity.

Appendix C

Matlab® Code for the E-Test

This appendix provides the MATLAB® code listing for my implementation of Székely and Rizzo's E-Test for equality of distributions with high dimensionality [51]. This code was applied in Section 4.2.3 to compute p-values for testing if the dynamical distributions of cellular automata rule spaces are effected by increaseing the genus of the topology.

The main function implements the k-sample E-test including the bootstrap re-sampling method of deriving Pr(E > e). The code for this function is given in Section C.1. The E-test code depends on the implementation of the k-sample E-statistic, which in turn depends on the implementation of the 2-sample E-statistic. The code for these dependencies are given in Sections C.2 and C.3 respectively.

The MATLAB® code in this appendix is based loosely on Székeley's own implementation in the statistical package R. Due to the method of memory allocation use by Székeley, the R version was not suitale for the data set sizes in my project. My MATLAB® implementation saves on memeory usage by recomputing Euclidean distances as required, rather than computing once and storing all results.

C.1 ksampleEtest.m

```
function [pval] = ksampleEtest(A,alpha)
% [PVAL] = KSAMPLEETEST(A,ALPHA) k-sample E-test
% for equality of equalitily of k empirical
% distributions.
k = length(A);
n = zeros(k,1);
d = zeros(k,1);
for i=1:k
    [n(i),d(i)] = size(A{i});
end
if sum(d ~= d(1)) > 0
    error('Distirbutions must have the same dimensionality.');
end
d = d(1);
% size of pooled data
N = sum(n);
% pooled data
X = zeros(N,d);
% offsets
m = ones(k,1);
for i=2:k
    m(i) = m(i-1)+n(i-1);
end
```

```
% copy sample data
for i=1:k
    X(m(i):(m(i)+n(i)-1),:) = A{i}(1:n(i),:);
end
% compute the observed test-statistic
E0 = estatk(A);
% determine number of resamples required
B = round(10^(abs(log10(alpha))))*10-1
E = zeros(B,1);
% derive bootstrap estmate of P(En <= t)
for b=1:B
    % generate a random permutation of the pooled sample
    perm = randperm(N);
    X(1:N,:) = X(perm,:);
    % generate new sample
    for i=1:k
        A{i}(1:n(i),:) = X(m(i):(m(i)+n(i)-1),:);
    end
    E(b) = estatk(A);
end
% P(En <= t) ~ 1/B sum(I(E0 <= t))
pval = (sum(E > E0)+1)/(B+1);
```

C.2 estatk.m

```
function [etest] = estatk(A)
% [ESTAT] = ESTATK(A) computes the k-sample
% test statistic for the multivariate E-test
% for equal distributions
k = length(A);
etest = 0;
for j=2:k
    for i=1:j-1
        etest = etest + estat2(A{i},A{j});
    end
end
```

C.3 estat2.m

```
function [etest] = estat2(X,Y)
% [ETEST] = ESTAT2(X,Y) computes the two-sample
% test statistic for the multivariate E-test for
% equal distibutions.
[n1,d1] = size(X);
[n2,d2] = size(Y);
if d1 ~= d2
    error('Distributions must have the same dimensionality.');
end
```

```
t1 = 0;
for i=1:n1
    X_i = repmat(X(i,:),[n2,1]);
    t1 = t1 + sum(sum((X_i - Y(1:n2,:)).^2,2).^(1/2));
end
t2 = 0;
for i=1:n1
    X_i = repmat(X(i,:),[n1,1]);
    t2 = t2 + sum(sum((X_i - X(1:n1,:)).^2,2).^(1/2));
end
t3 = 0;
for i=1:n2
    Y_i = repmat(Y(i,:),[n2,1]);
    t3 = t3 + sum(sum((Y_i - Y(1:n2,:)).^2,2).^(1/2));
end
etest = ((n1*n2)/(n1+n2))* ...
 ((2.0/(n1*n2))*t1 - (1.0/n1*n1)*t2 - (1.0/n2*n2)*t3);
```

Appendix D

Entropy Measure Comparisons for Selected Rules

/

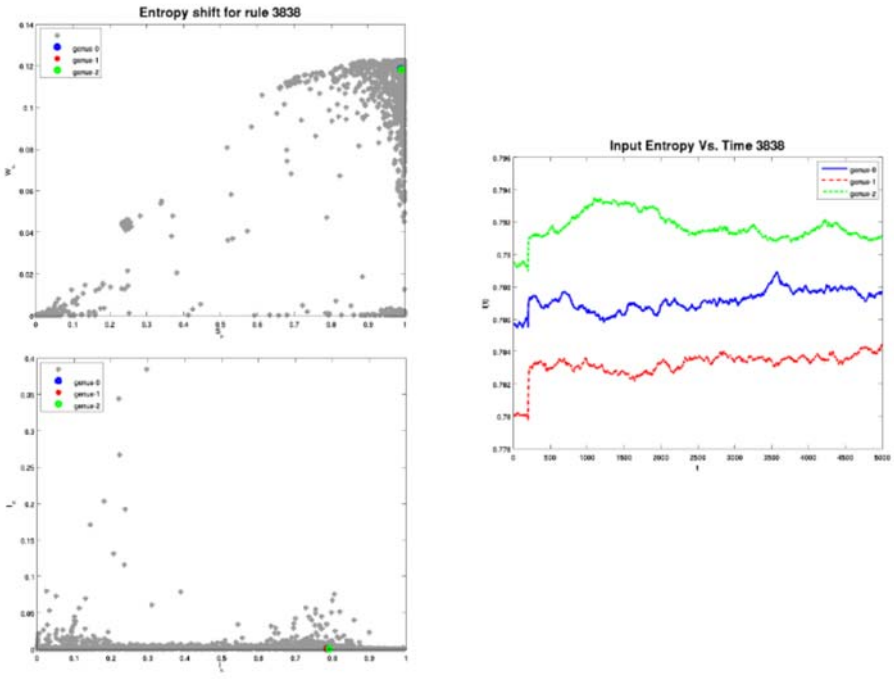

Figure D.1: Entropy Shift in life rule 3838.

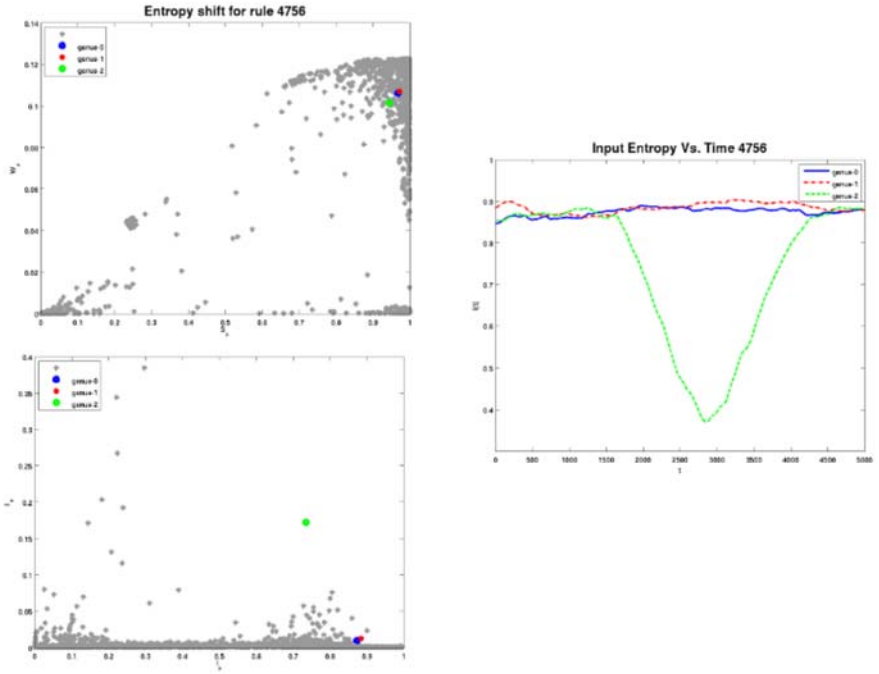

Figure D.2: Entropy Shift in life rule 4756.

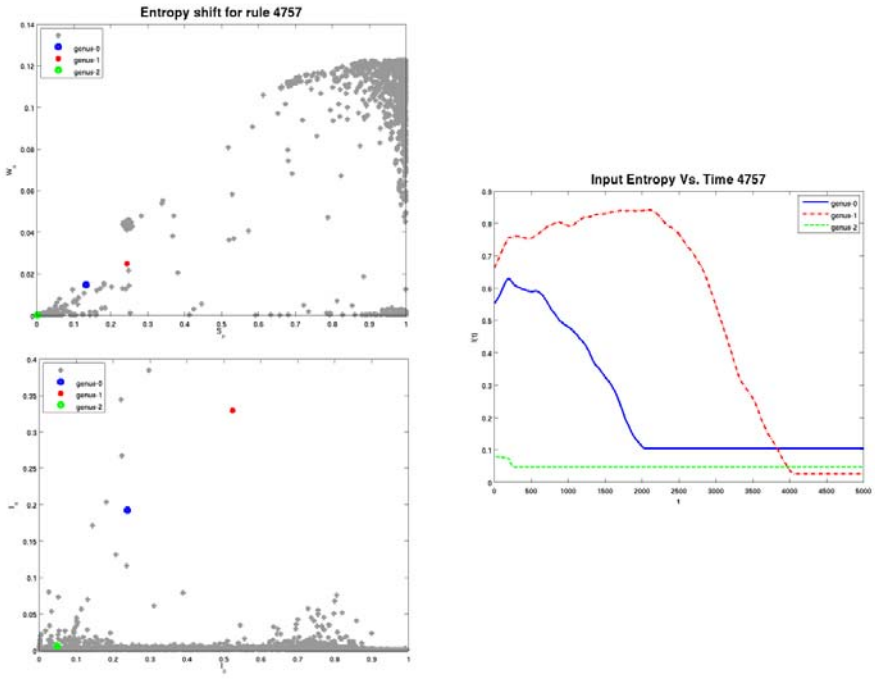

Figure D.3: Entropy Shift in life rule 4757.

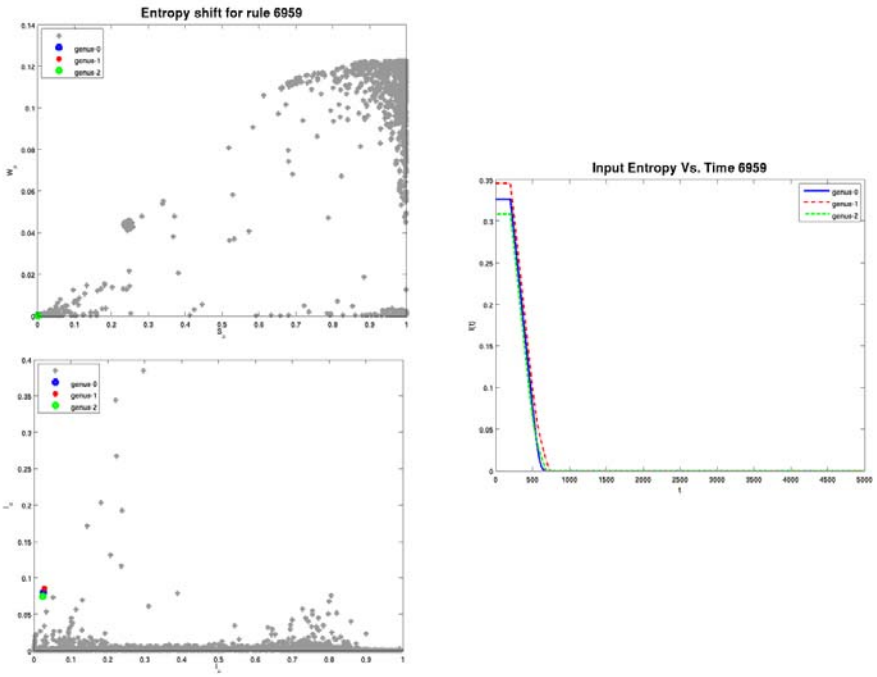

Figure D.4: Entropy Shift in life rule 6959.

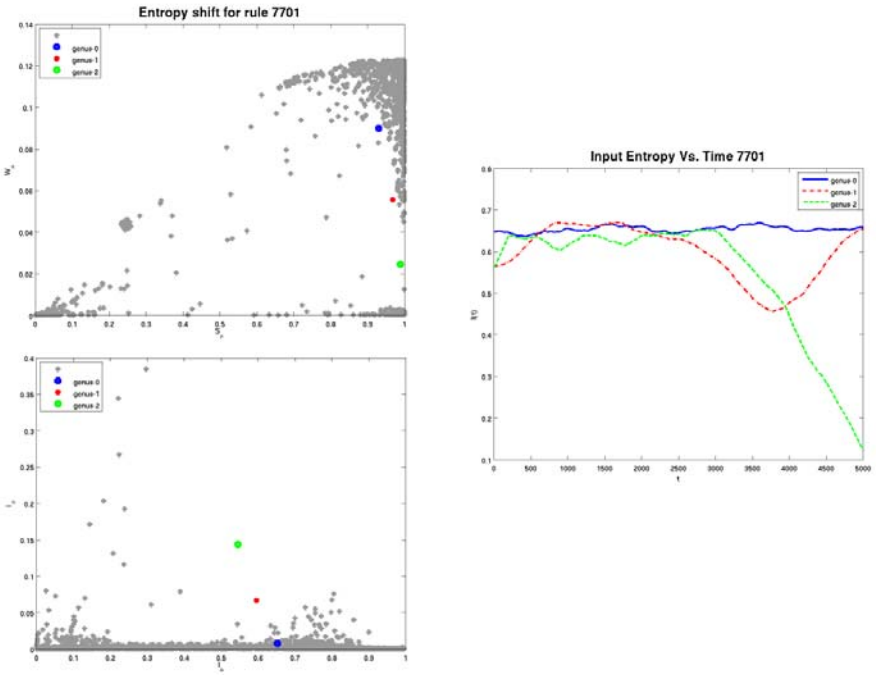

Figure D.5: Entropy Shift in life rule 7701.

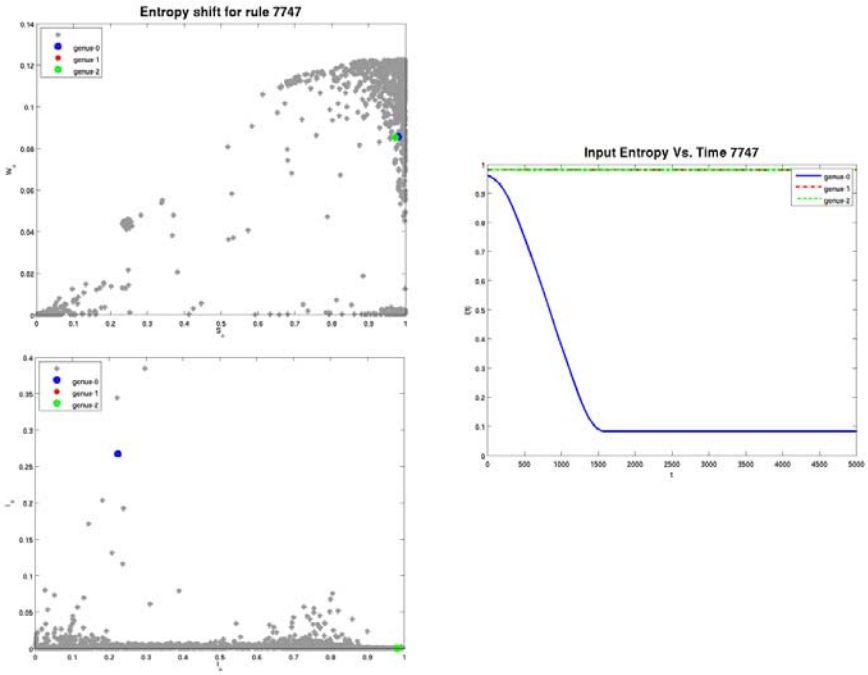

Figure D.6: Entropy Shift in life rule 7747.

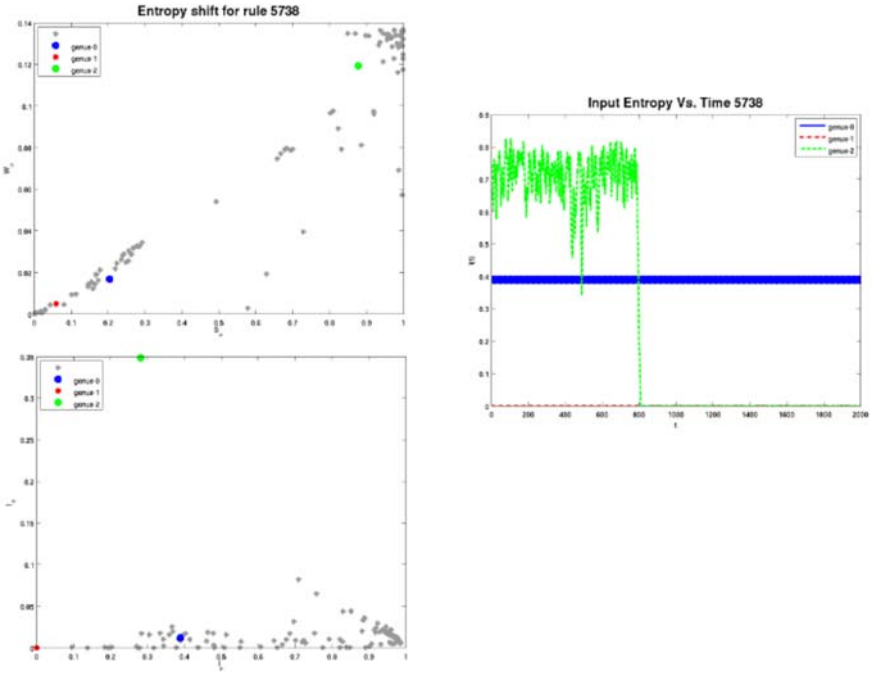

Figure D.7: Entropy Shift in outer-totalistic rule 5738 with $\mathsf{N} = 80$.

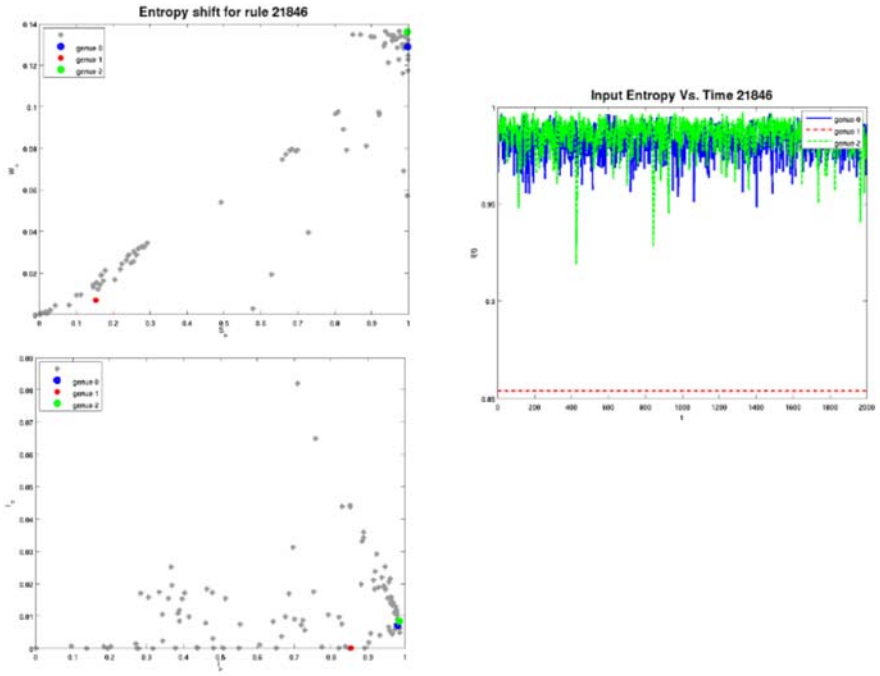

Figure D.8: Entropy Shift in outer-totalistic rule 21846 with $\mathsf{N} = 80$.

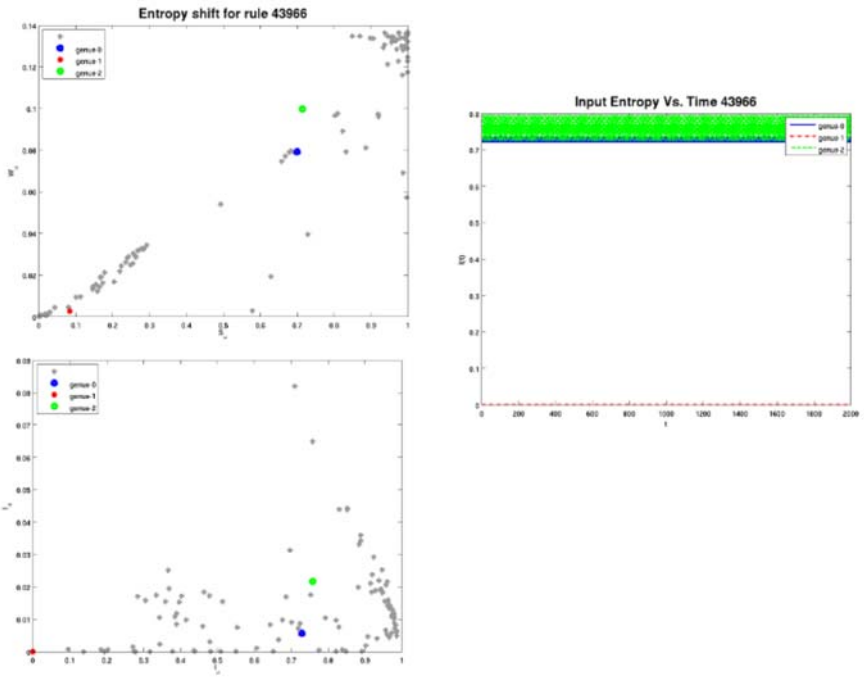

Figure D.9: Entropy Shift in outer-totalistic rule 43966 with $\mathsf{N} = 80$.

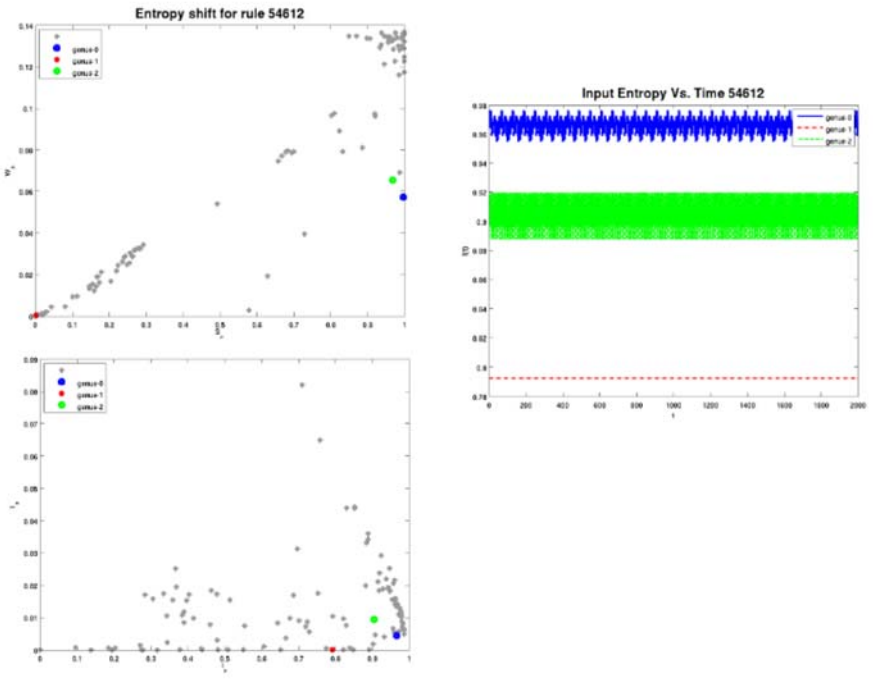

Figure D.10: Entropy Shift in outer-totalistic rule 54612 with $\mathsf{N} = 80$.

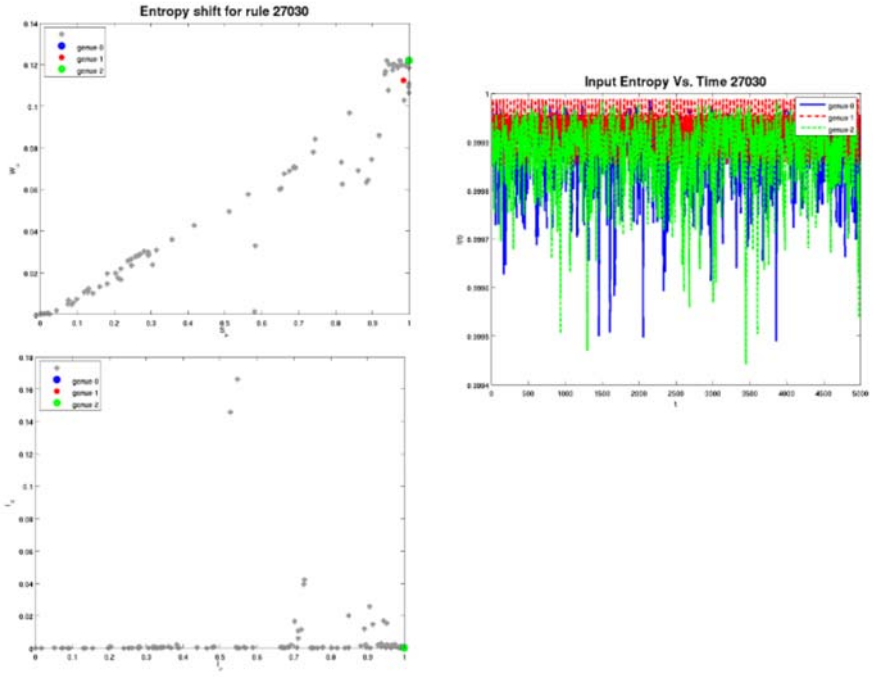

Figure D.11: Entropy Shift in outer-totalistic rule 27030 with $\mathsf{N} = 1280$.

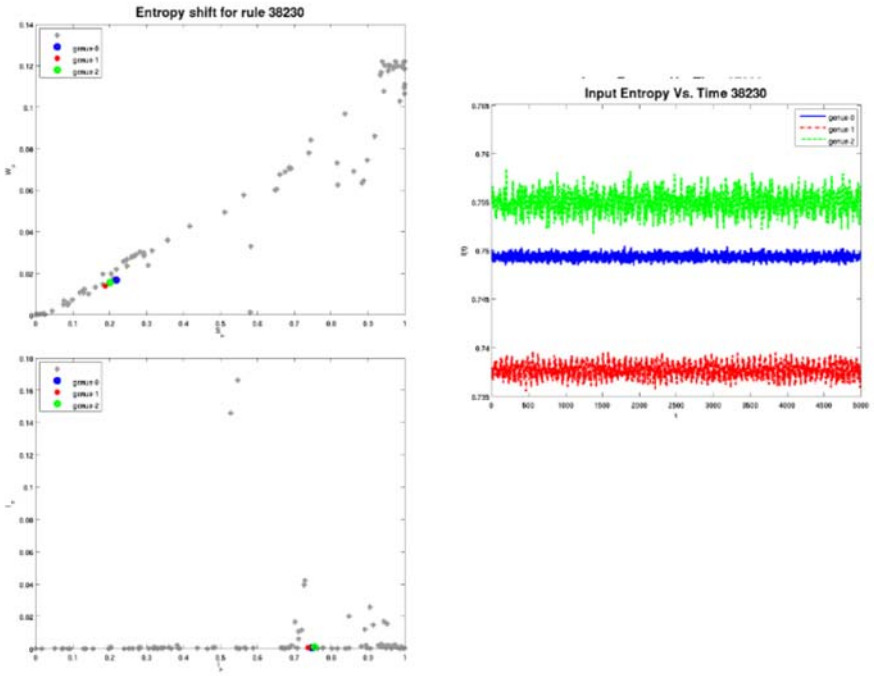

Figure D.12: Entropy Shift in outer-totalistic rule 38320 with $\mathsf{N} = 1280$.

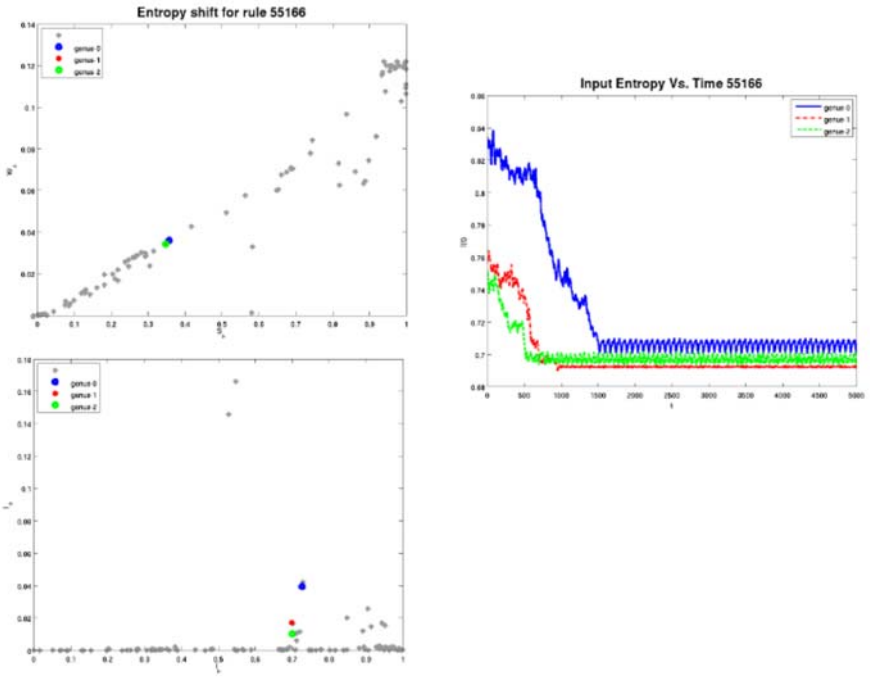

Figure D.13: Entropy Shift in outer-totalistic rule 55166 with $\mathsf{N} = 1280$.

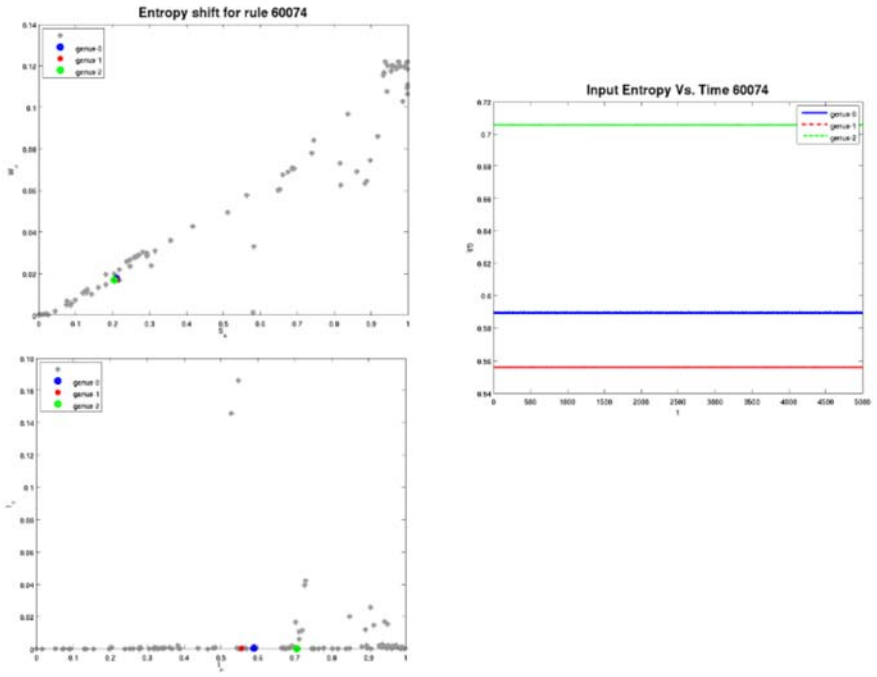

Figure D.14: Entropy Shift in outer-totalistic rule 60074 with $\mathsf{N} = 1280$.

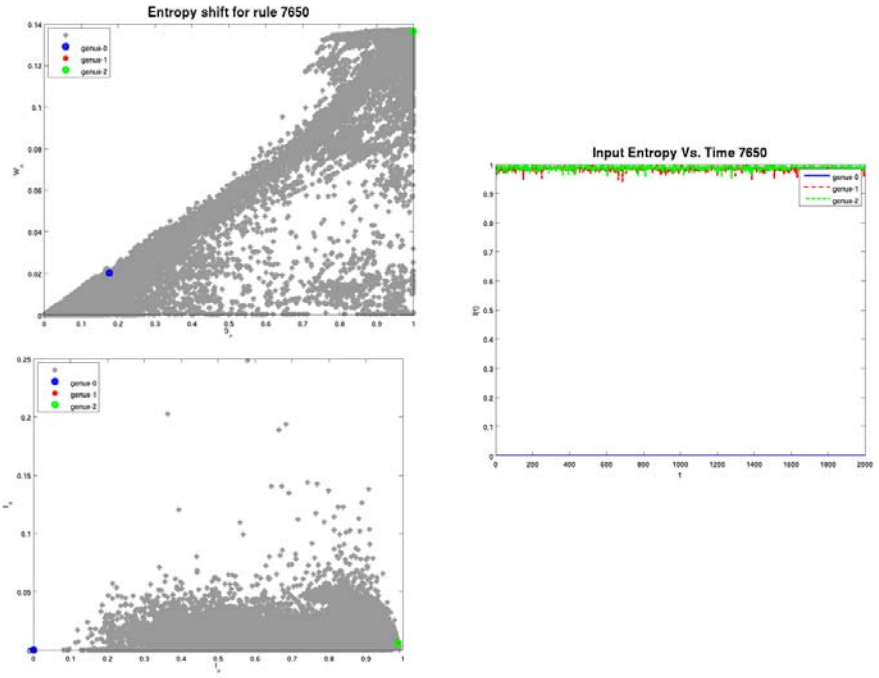

Figure D.15: Entropy Shift in rule 7650 with $\mathsf{N} = 80$.

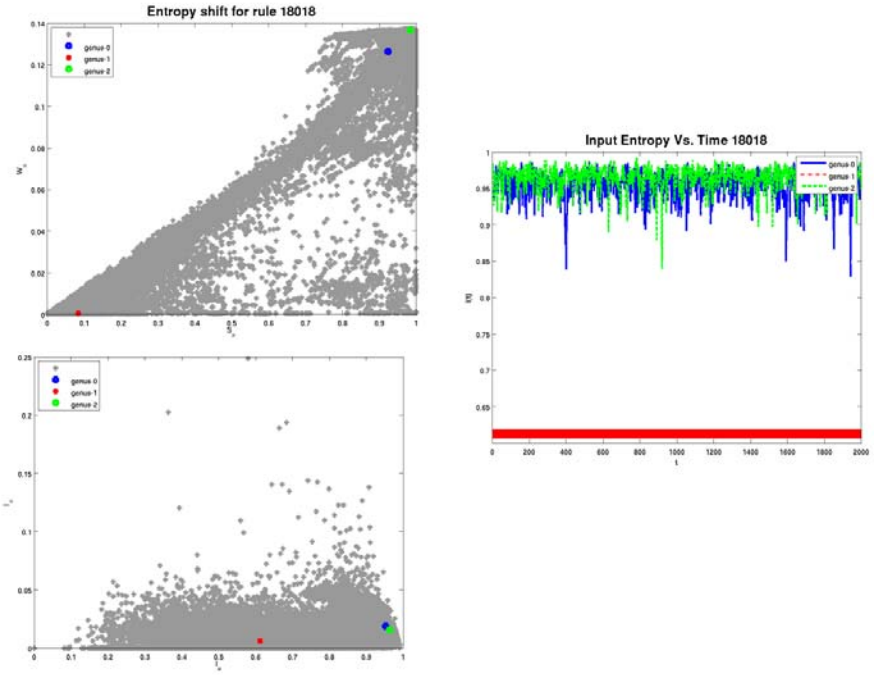

Figure D.16: Entropy Shift in rule 18018 with $\mathsf{N} = 80$.

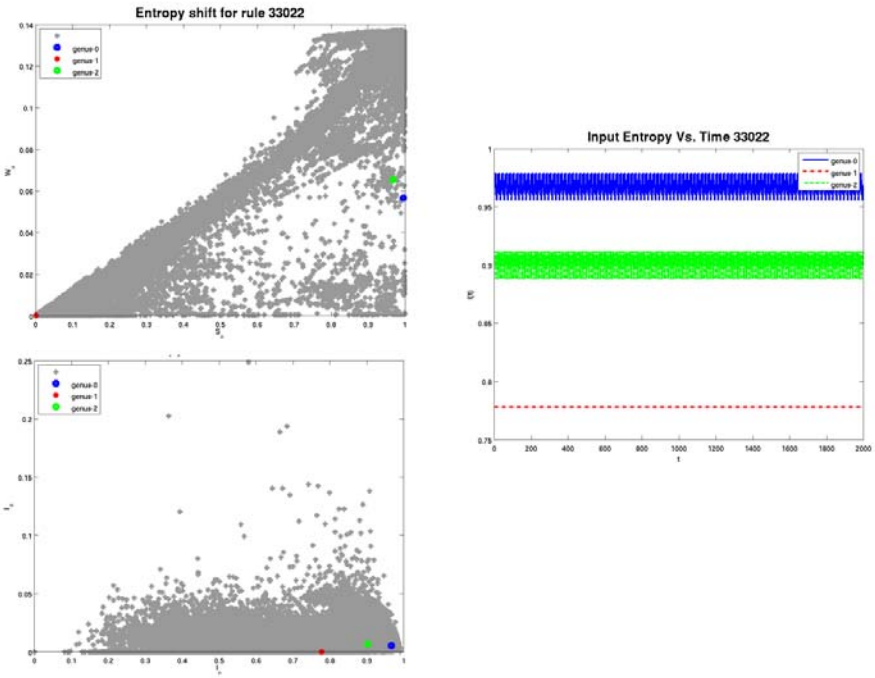

Figure D.17: Entropy Shift in rule 33022 with $\mathsf{N} = 80$.

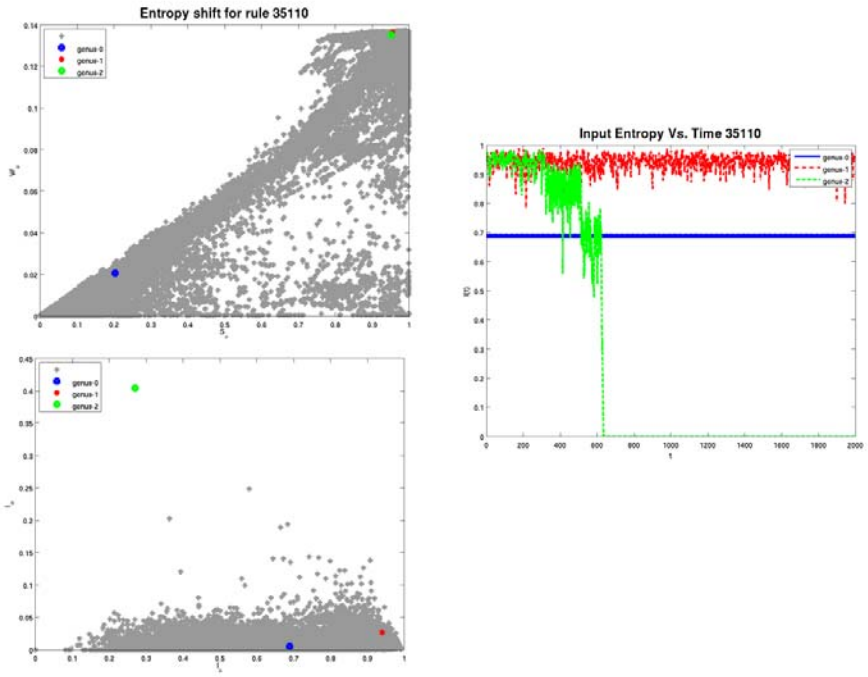

Figure D.18: Entropy Shift in rule 35110 with $\mathsf{N} = 80$.

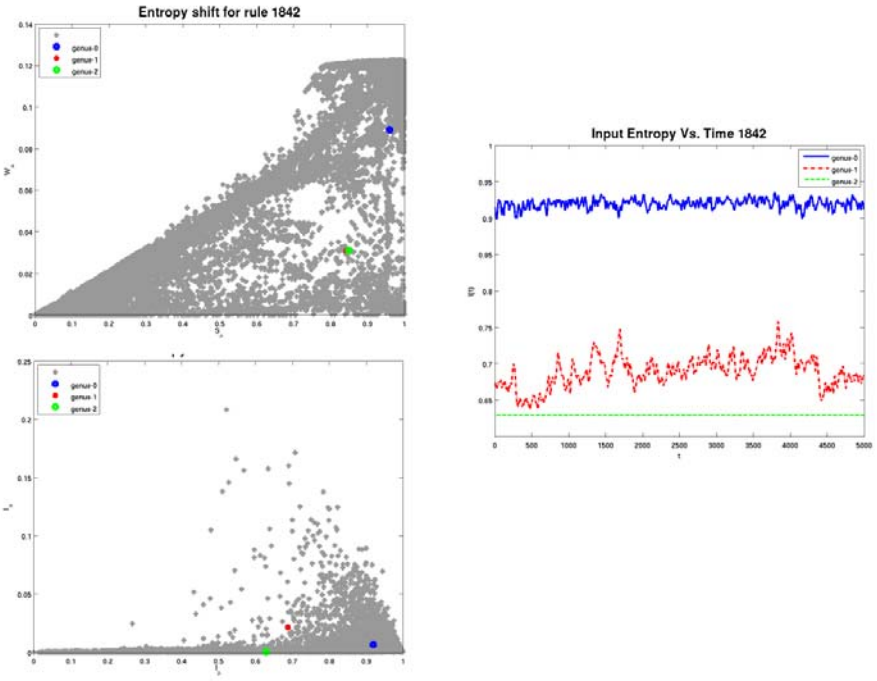

Figure D.19: Entropy Shift in rule 1842 with $\mathsf{N} = 1280$.

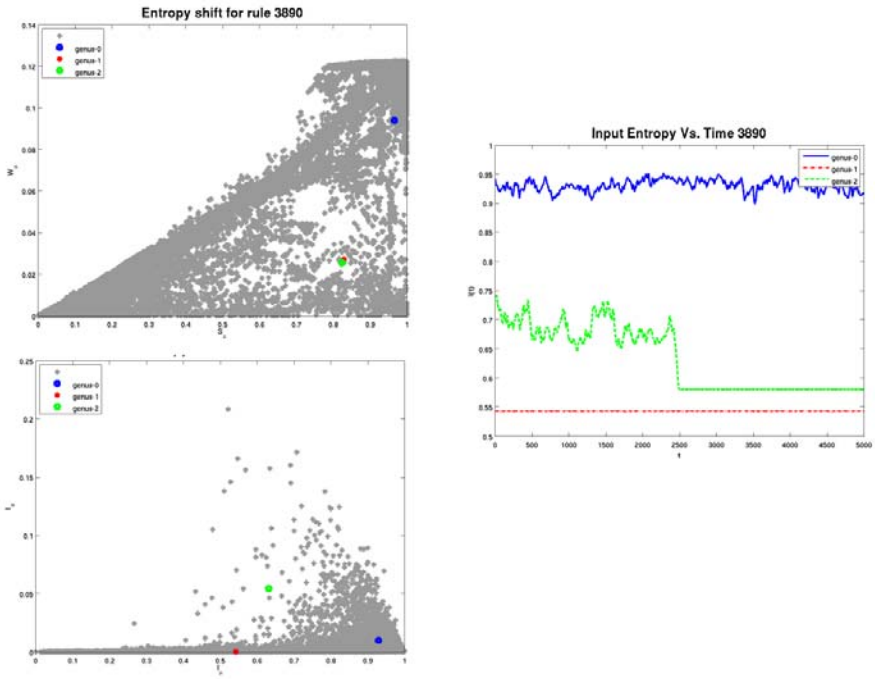

Figure D.20: Entropy Shift in rule 3890 with $\mathsf{N} = 1280$.

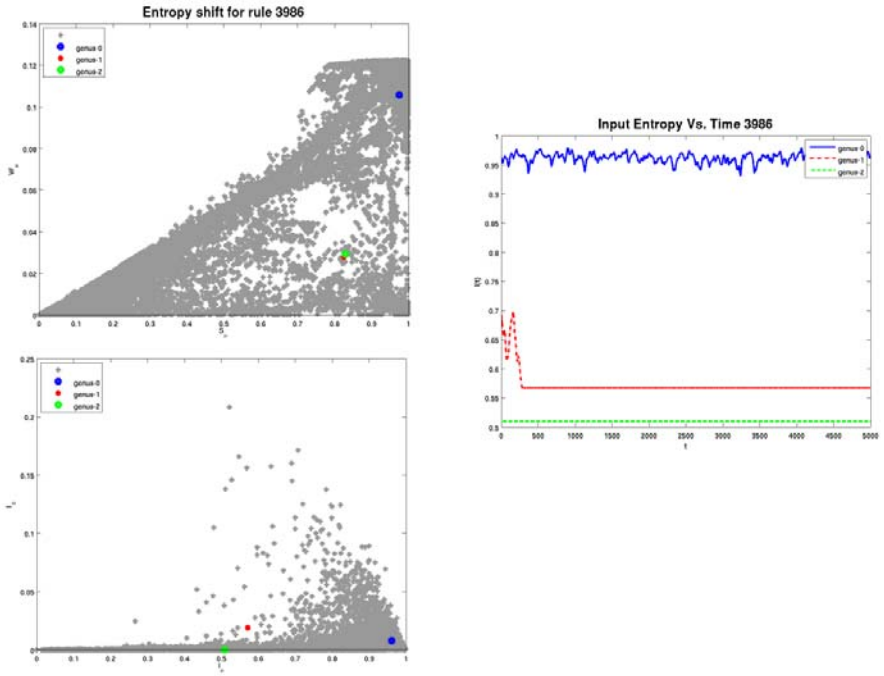

Figure D.21: Entropy Shift in rule 3986 with $\mathsf{N} = 1280$.

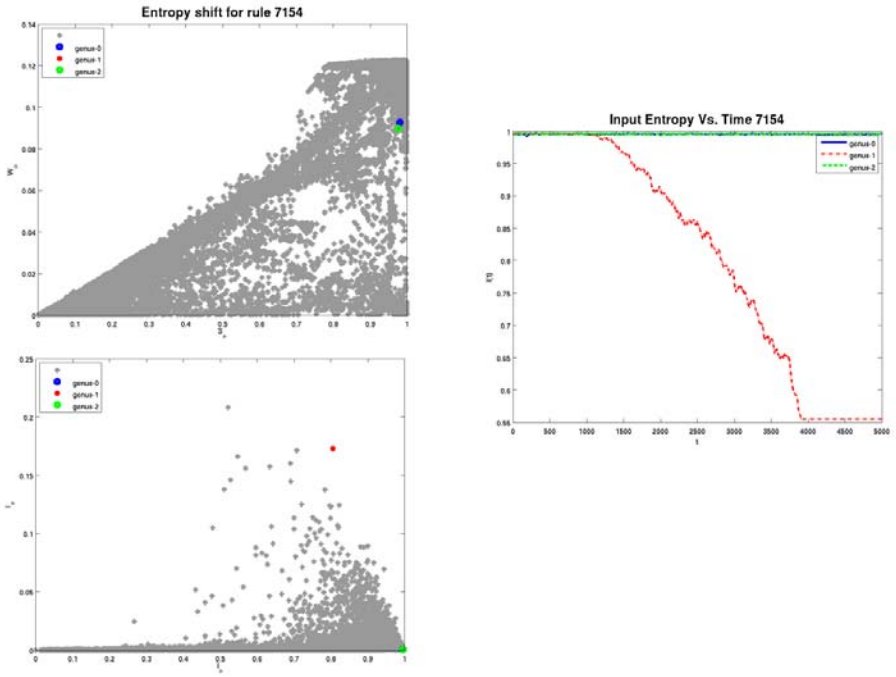

Figure D.22: Entropy Shift in rule 7154 with $\mathsf{N} = 1280$.

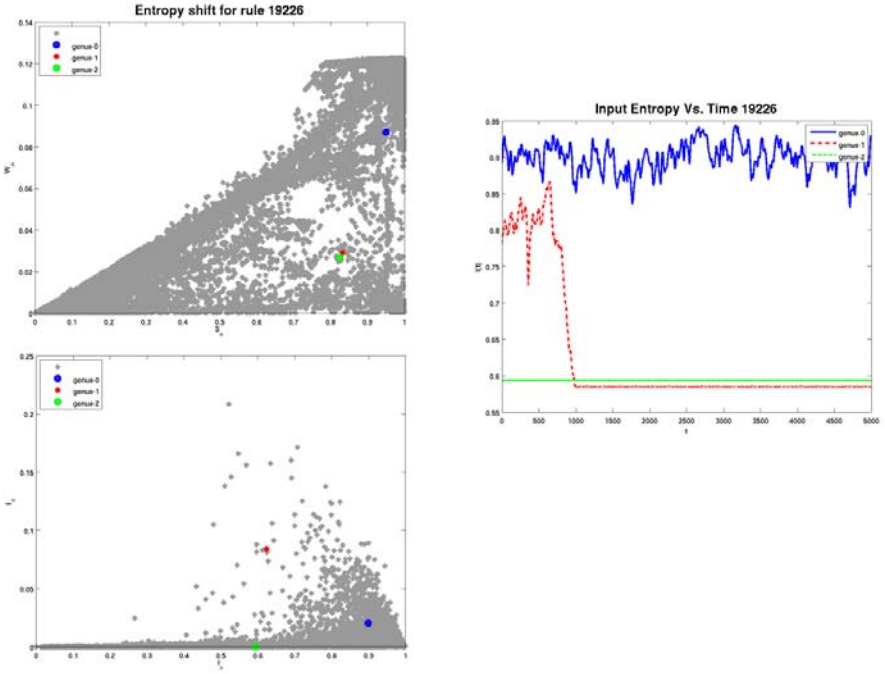

Figure D.23: Entropy Shift in rule 19226 with $\mathsf{N} = 1280$.

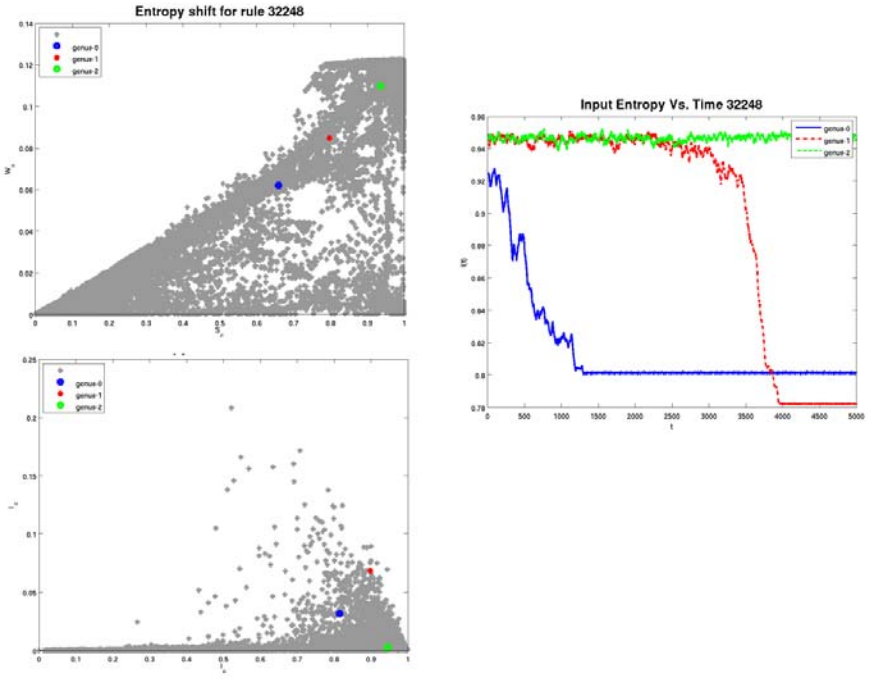

Figure D.24: Entropy Shift in rule 32248 with $\mathsf{N} = 1280$.

D. ENTROPY MEASURE COMPARISONS FOR SELECTED RULES

References

[1] A. Adamatzky, G. J. Martinez, and J. C. S. T. Mora. Phenomenology of reation-diffusion binary-state cellular automata. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, **16**(10):2985–3006, 2006. 24, 25

[2] M. H. Afshar, M. Shahidi, M. Rohani, and M. Sargolzaei. Application of cellular automata to sewer network opimization problems. *Scientia Iranica*, **18**(3):304–312, 2011. 25

[3] S. Amoroso and G. Cooper. The garden-of-eden theorem for finite configurations. *Proceedings of the American Mathematical Society*, **26**:158–164, 1970. 20

[4] S. Bandini, G. Mauri, and R. Serra. Cellular automata: From a theoretical parallel computational model to its application to complex systems. *Parallel Computing*, **27**(5):539–553, 2001. 3, 9, 11, 19, 24, 25

[5] A. Bandyopadhyay, R. Pati, S. Sahu, F. Peper, and D. Fujita. Massively parallel computing on an organic molecular layer. *Nature Physics*, **6**(5):369–375, 2010. 1, 25

[6] C. Barret, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Strearns, and M. Thakur. Predecessor existence problems for finite dicrete dynamical systems. *Theoretical Computer Science*, **386**:3–37, 2007. 20

[7] C. Bays. Cellular automata in the triangular tessellation. *Complex Systems*, **8**:127–150, 1994. 5, 6, 15, 35, 36, 45, 46, 61

[8] M. A. Bedau. Artificial life: organization, adaptation and complexity from the bottem up. *Trends in Cognitive Sciences*, **7**(11):505–512, 2003. 24

[9] C. R. Calidonna, S. Di Gregorio, and M. Mango Furnari. Mapping applications of cellular automata into applications of cellular automata networks. *Computer Physics Communications*, **147**:724–728, 2002. 3, 25

[10] L. O. Chua and L. Yang. Cellular neural networks. *IEEE Transactions on Circuits and Systems*, **35**(10):1257–1272, 1988. 25

[11] M. Cook. Universality in elementary cellular automata. *Complex Systems*, **15**:1–40, 2004. 3, 10, 11, 18, 19

[12] M. Cook. A concrete view of rule 110 computation. In *The Complexity of Simple Programs*, pages 31–55, 2008. 18

[13] B. Efron and R. Tibshirani. *An introduction to the bootstrap*, **57** of *Monographs on statistics and applied probability*. Chapman & Hall, New York, 1993. 48

[14] D. Eppstein. Growth and decay in life-like cellular automata. In A. Adamatzky, editor, *Game of Life Cellular Automata*, pages 71–98. Springer-Verlag, 2010. 16

[15] F. Fasano and A. Franceschini. A multidimensional version of the kolmogorov-smirnov test. *Monthly Notices of the Royal Astronomical Society*, **225**:155–170, 1987. 31, 47

[16] N. Fatès. Critical phenomena in cellular automata: Perturbing the update, the transitions, the topology. *Acta Physica Polonica B, Proceedings Supplement*, **3**(2):315–325, 2010. 2, 4, 5, 27, 28, 29, 83, 89

[17] W. Feller. On the kolmogorov-smirnov limit theorems for empirical distributions. *Annals of Mathematical Statistics*, **19**:177–189, 1948. 47

[18] M. Gardner. Mathematical games: The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, **223**:120–123, 1970. 10, 18

[19] K. Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für Mathematik und Physik*, **38**:173–198, 1931. 18

[20] J. Gravner, G. Gliner, and M. Pelfray. Replication in one-dimensional cellular automata. *Physica D*, **240**:1460–1474, 2011. 24

[21] S. Hawking and L. Mlodinow. *The Grand Design*. Bantam Books, an imprint of The Random House Publishing Group, a division of Random House, Inc., New York, 1 edition, 2010. 2, 10

[22] D. R. Hofstadter. *Gödel,Escher,Bach: an eternal golden braid*. Basic Books, Inc., New York, 1979. 18

[23] G.'t Hooft. Duality between a deterministic cellular automaton and a bosonic quantum field theory in 1+1 dimensions. *Foundations of Physics*, **43**(5):597–614, 2013. 2, 10

[24] E. Jen. Enumeration of preimages in cellular automata. *Complex Systems*, **3**:421–456, 1989. 20

[25] D. H. Jones, R. McWilliam, and A. Purvis. Designing convergent cellular automata. *BioSystems*, **96**(1):80–85, 2009. 24

[26] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, **22**:437–467, 1969. 10, 12, 13, 19, 21, 24, 31, 44

[27] P. Kůrka. Languages, equicontinuity and attractors in cellular automata. *Ergodic Theory and Dynamical Systems*, **17**(2):417–433, 1997. 15, 16, 30

[28] C. G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, **42**:12–37, 1990. 3, 10, 11, 14, 15, 16, 19, 20, 24, 30, 31, 39, 41, 44, 53

[29] C. G. Langton. *Artificial Life*. Addison-Wesley, 1992. 24

[30] D. G. Mallet and L. G. De Pillis. A cellular automata model of tumor-immune system interactions. *Journal of Theoretical Biology*, **239**(3):334–350, 2006. 24

[31] N. Margolus. Cam-8: a computer architecture based on cellular automata. 1993. 25

[32] C. Marr and M.-T. Hütt. Topology regulates pattern formation capacity of binary cellular automata on graphs. *Physica A*, **354**:641–662, 2005. 2, 4, 5, 11, 13, 15, 19, 22, 23, 27, 28, 29, 30, 31, 39, 41, 53, 57, 83

[33] C. Marr and M.-T. Hütt. Outer-totalistic cellular automata on graphs. *Physics Letters A*, **373**(5):546–549, 2009. 2, 4, 5, 13, 15, 19, 22, 23, 27, 28, 29, 39, 41, 53, 57, 83, 90

[34] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 1993. 20

[35] N. H. Packard. Adaptation toward the edge of chaos. In *Dynamical patterns in complex systems*, pages 293–301. World Scientific, 1988. 20, 24

[36] J. A. Peacock. Two-dimensional goodness-of-fit testing in astromomy. *Monthly Notices of the Royal Astronomical Society*, **202**:615–627, 1983. 31, 47

[37] P. R. Rosenbaum. An exact distribution-free test comparing two multivariate distributions based on adjacency. *Journal of the Royal Statistical Society B*, **67**(4):515–530, 2005. 31, 47

[38] P. L. Rosin. Image processing using 3-state cellular automata. *Computer Vision and Image Understanding*, **114**(7):790–802, 2010. 25

[39] M. Sablik and G. Theyssier. Topological dynamics of cellular automata: Dimension matters. *Theory of Computing Systems*, **48**(3):693–714, 2010. 16

[40] J. B. Salem and S. Wolfram. Thermodynamics and hydrodynamics with cellular automata. *Thinking Machines Corporation technical report*, 1985. 24

[41] S. Seif. Constrained eden. *Complex Systems*, **18**:279–385, 2009. 16, 20

[42] R. Serra and M. Villani. Perturbing the regular topology of cellular automata: Implications for the dynamics. In S. Bandini, B. Chopard, and M. Tomassini, editors, *The 5th International Conference on Cellular Automata for Research and Industry*, **2493** of *Lecture Notes in Computer Science*, pages 168–177, Berlin Heidelberg, 2002. Springer-Verlag. 2, 4, 27, 83, 89

[43] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, **27**(3):379–423, 1948. 22

[44] P. L. Shick. *Topology: Point-Set and Geometric*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2011. 25

[45] G. Ch. Sirakoulis, I. Karafyllidis, and A. Thanailakis. A cellular automaton methodology for the simulation of integrated circuit fabrication process. *Future Generation Computer Systems*, **18**:639–657, 2002. 25

[46] N. Smirnov. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. *Bulletin Mathématique de l'Université de Moscou*, **2**, 1939. 47

[47] D. Stauffer. Cellular automata and hydrodynamic applications. *Pysica Scripta*, **35**:66–70, 1991. 24

[48] D. Stevens, D. Scott, and J. Silk. Microwave background anisotropy in a toroidal universe. *Phyiscal Review Letters*, **71**(1):20–23, 1993. 92

[49] K. Sutner. De bruijn graphs and linear cellular automata. *Complex Systems*, **5**:19–30, 1991. 20

[50] K. Sutner. On the computational complexity of finite cellular automata. *Journal of Computer and System Sciences*, **50**:87–97, 1995. 20

[51] G. J. Szēkely and M. L. Rizzo. Testing for equal distributions in high dimension. In *InterStat*, 2004. 31, 47, 48, 133

[52] G. J. Szēkely and M. L. Rizzo. A new test of multivariate normality. *Journal of Multivariate Analysis*, **93**:58–80, 2005. 31, 47

[53] T. Toffoli. Cam: A high-performance cellular-automaton machine. *Physica D*, **10**(1):195–204, 1984. 25

[54] T. Toffoli and N. Margolus. *Cellular automata machines*. MIT Press, 1987. 25

[55] M. Tomassini. Generalized automata networks. *Lecture Notes in Computer Science*, **4173**:14–28, 2006. 4, 11, 13, 27, 28, 29, 83

[56] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. In *The proceedings of the London Mathematical Society*, **42** of *2*, 1936. 18

[57] J. Ventrella. Glider dynamics on the sphere: Exploring cellular automata on geodesic grids. *Journal of Cellular Automata*, **6**(2):245–256, 2011. 4, 5, 6, 11, 28, 29, 36, 80, 85, 86, 87, 89, 91

[58] N. Vlassopoulos, N. Fatēs, H. Berry, and B. Girau. Large-scale simulations on fpgas: Finding the asymptotic critical threshold of the greenberg-hastings cellular automata. *Journal of Cellular Automata*, **7**(1):5–29, 2012. 25

[59] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana and London, 1966. 10, 24

[60] R. T. Wainwright. Life is universal! In *The 7th Conference on Winter Cimulation*, **2**, pages 449–460. ACM, 1974. 2, 3, 5, 6, 10, 18, 36

[61] S. Wolfram. Cellular automata. *Los Alamos Science*, **9**:2–21, 1983. 2, 9, 11, 15, 16, 19

[62] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, **55**:601–644, 1983. 3, 11, 14, 15, 16, 21, 22, 24, 28, 41

[63] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, **10**:1–35, 1984. 3, 11, 15, 16, 28, 41, 58

[64] S. Wolfram. Complex systems theory. In *Founding Workshops of the Santa Fe Institute*, pages 183–189. Addison-Wesley, 1985. 1, 3, 10, 11, 24, 25, 83

[65] S. Wolfram. Twenty problems in the theory of cellular automata. *Physica Scripta*, **T9**:170–183, 1985. 11, 24, 25

[66] S. Wolfram. Undecidability and intractability in theoretical physics. *Physical Review Letters*, **54**(8):735–738, 1985. 2, 11, 19, 21, 22, 24

[67] S. Wolfram. Minimal cellular automaton approximations to continuum systems. pages 329–358, 1986. 24

[68] S. Wolfram. Tables of cellular automaton properties. *Theory and Applications of Cellular Automata*, pages 485–557, 1986. 19, 21, 44, 66

[69] S. Wolfram. Cellular automaton supercomputing. In R. B. Wilhelmson, editor, *High-Speed Computing: Scientific Applications and Algorithm Design*. University of Illinois Press, 1988. 19, 25

[70] S. Wolfram. *A New Kind of Science*. Wolfram Media, Inc., Champaign, IL, 2002. 2, 10, 11, 24

[71] A. Wuensche. The ghost in the machine: Basins of attraction of random boolean networks. In *Artificial Life III*, 1993. 12, 13, 19, 23

[72] A. Wuensche. Discrete dynamical networks and their attractor basins. In *Complex Systems '98*, 1998. 3, 19, 20, 21

[73] A. Wuensche. Genomic regulation modeled as a network with basins of attraction. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klien, editors, *Pacific Symposium on Biocomputing '98*, pages 89–102. World Scientific, 1998. 13, 24

[74] A. Wuensche. Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the z parameter. *Complexity*, **4**(3):47–66, 1999. 3, 19, 20, 21, 23, 28, 30, 31, 39, 41, 42, 44, 53, 57

[75] A. Wuensche. Glider dynamics in 3-value hexagonal cellular automata: the beehive rule. *International Journal of Unconventional Computing*, **1**(4):375–398, 2005. 24

[76] A. Wuensche. Complex and chaotic dynamics, basins of attraction, and memory in discrete networks. *Acta Physica Polonica B*, **3**(2):463–478, 2010. 1, 10, 15, 16, 21, 24, 25, 28, 31, 44, 57, 66, 84

[77] A. Wuensche. *Exploring Discrete Dynamics*. Luniver Press, Frome, United Kingdom, 2011. 21, 23

[78] A. Wuensche and M. Lesser . *The global dynamics of cellular automata*, **1**. Addison-Wesley, Reading, MA, 1992. 13, 19, 20, 21, 23