1-1-2011

Information Processing in Two-Dimensional Cellular Automata

Martin Cenek
Portland State University

Information Processing in Two-Dimensional Cellular Automata

by

Martin Cenek

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Computer Science

Dissertation Committee:
Melanie Mitchell, Chair
Bart Massey
Cynthia Brown
James G. Hook
Dan Hammerstrom

Portland State University

ABSTRACT

Cellular automata (CA) have been widely used as idealized models of spatially-extended dynamical systems and as models of massively parallel distributed computation devices. Despite their wide range of applications and the fact that CA are capable of universal computation (under particular constraints), the full potential of these models is unrealized to-date. This is for two reasons: (1) the absence of a programming paradigm to control these models to solve a given problem and (2) the lack of understanding of how these models compute a given task. This work addresses the notion of computation in two-dimensional cellular automata.

Solutions using a decentralized parallel model of computation require information processing on a global level. CA have been used to solve the so-called density (or majority) classification task that requires a system-wide coordination of cells. To better understand and challenge the ability of CA to solve problems, I define, solve, and analyze novel tasks that require solutions with global information processing mechanisms.

The ability of CA to perform parallel, collective computation is attributed to the complex pattern-forming system behavior. I further develop the *computational mechanics* framework to study the mechanism of collective computation in two-dimensional cellular automata. I define several approaches to automatically identify the spatiotemporal structures with information content. Finally, I demonstrate why an accurate model of information processing in two-dimensional cellular automata cannot be constructed from the space-time behavior of these structures.

DEDICATION

TO MY PARENTS, CARRIE, SUE and LARRY.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Dr. Melanie Mitchell for everything she has ever done for me. She has been a great mentor and advisor with overabundance of inspiration, patience, guidance, and encouragement. It is my honor to be her student. I would also like to thank to everybody who advised along the way; all of my advisors, for their input and their willingness to listen (to all the good and the bad). Dr. Bart Massey and Dr. Cynthia Brown always went the extra mile for me. Dr. Christof Teuscher always had advice and thoughts when I needed them.

This work has been funded by the Center on Functional Engineered Nano Architectonics (FENA), through the Focus Center Research Program of the Semiconductor Industry Association. The remainder of my support was from the Computer Science Department. Thank you for the generous support.

Working with the Learning and Adaptive Systems group was always a pleasure. Ralf Juengling is a super-star. Thank you not only for the gift of 'lush', but for being a good friend who will never be forgotten. Dr. Manuel Marques-Pita as a collaborator on this work, and Mick Thomure and Will Landecker who were always willing listeners. Dr. Alexander Weber of Lawrence Berkeley Labs for his friendship, and putting my work into perspective. Everyone from CSGCS, but most of all Dr. Chuan-kai Lin, Rashawn Knapp, Dr. Kathryn Mohror. Finally, I would like to thank to all of my friends and family who believed in me and supported me on this journey.

CONTENTS

[1] **Portions of this chapter were adapted from [76]**
[2] **Portions of this chapter were adapted from [76]**

LIST OF TABLES

LIST OF FIGURES

CONTENTS

Chapter 1

INTRODUCTION

In recent years, the study of complex systems, self-organized living organisms, real-world networks and adaptive systems have attracted an increasing amount of research interest. These systems, though vastly different, also display important commonalities. Each system is decentralized and composed of simple, locally connected components. However, these systems are capable of demonstrating complex system-wide behavior. The capacity to perform complex behavior could be due to information being communicated through the network by a signal propagating mechanism and the occurrence of some form of information processing among these signals.

It is difficult to understand how these complex systems process information. In order to understand how these systems perform tasks that require collective cooperation, we need to both study the individual components and investigate the system as a whole. Simplifying these systems and introducing constraints will yield a model that is easier to study, manipulate and apply to real-life systems. A cellular automaton (CA) could be considered one such model.

The spatio-temporal behavior of a CA performing a computation shows complex, non-linear, and non-intuitive behavior. The analysis of a system's global collective behavior cannot be easily inferred from the interactions among locally connected components. In order to understand how these systems perform a given task, first we need to identify the information carrying signals, second we need to capture how the location of these information carrying channels changes over time, and finally combine these analyses into a model that captures the mechanisms of

information processing.

Chapter 2 introduces the reader to the background topics of cellular automata, the early models of information processing in CA, and the computation in one- and two-dimensional CA. The computational tasks for CA are defined in Chapter 3 along with their potential applications. The methods used to solve these tasks are described in Chapter 4. Chapter 5 presents the results and analysis of GA evolved rules for various tasks that require a system-wide cooperation of the cells in two-dimensional cellular automata (2DCA). Chapter 7 introduces methods for the automatic identification of information carrying structures in 2DCA. Preliminary work on modeling the dynamic properties of the information carrying structures identified by these methods is presented in Chapter 8. Related research areas are summarized in Chapter 9 while Chapter 10 contains the final remarks and the outline of future research.

1.1 MOTIVATION

In 1965, Intel co-founder Gordon Moore predicted that the number of transistors on a processor would double every 18 months. This trend, also known as Moore's law, still holds true today; in 2010 Intel's Tukwila processor consisted of over 2 billion transistors [117]. However, the miniaturization of silicon-based devices is nearing its physical limits. In order to sustain the rate of innovation, we need smaller, faster, and more energy-efficient architectures. To address these demands, new research fields such as molecular electronics, spintronics, nano-robotics, and computational nano-fabrics were established.

The decreasing component size and increasing clock frequencies create new challenges for the semiconductor industry. For example, a computer's central processing unit (CPU) has many units that serve different computational functions (the Arithmetic Logic Unit consists of add, subtract, shift, rotate, divide, multiply, flag blocks). The microprocessor components are a product of a divide and conquer

design approach where the component's function is sub-divided into smaller tasks that are then solved independently. Although the reductionistic design approach will result in a component design according to desired specifications, it might not guarantee the desired performance of a system after the components are integrated. The inter-operation of the functional blocks depends on the exchange of data, instructions, and signals. The high frequency clock rate does not guarantee that the information will arrive from the source to the destination module in a single clock cycle. Since multiple clock cycles might be needed for information to reach its destination, the timing and ordering of tasks in the destination module might be different than in a single clock cycle scenario. Although the system integration aims to avoid race conditions, unforeseen CPU design side-effects will arise from the complex CPU behavior.

One way of troubleshooting unwanted system behavior is to view the design and operation of a CPU as a dynamic complex system. This view of a proposed architecture will allow for behavior analysis beyond the component level. The flow of information, the changes of the thermal signature, the occurrence of signal interactions, and the development of noise can be viewed as system-wide patterns. Understanding where these patterns occur, how their location changes in time, and what happens when these patterns interact will help with the design, integration and enhancement of system performance.

The microprocessor design field is one example where wider view of design and analysis could prove beneficial. Mobile device processing is another area that has not yet harnessed the power of collaborative (emergent) behavior among the system components. A hybrid decentralized network of peer-to-peer mobile devices, such as cell phones in cars, could use built-in accelerometers to "sense" impending traffic congestion. Similarly, a massive network of sensors could be used to spontaneously discover environmental changes such as tsunamis or geodetic events. Finally, advances in the field of nano-technology will result in a new generation of

electronic devices. These architectures will consist of inherently parallel, potentially faulty, locally connected, and decentralized components. Devices consisting of peta-order nano-switches and higher are feasible due to advances in hardware nano-fabrications.

So, what is stopping us from manufacturing nano-scale devices and building applications that solve problems by global, collaborative behavior among its components? Two reasons: we don't know in general how to control these devices to solve a problem, and we don't fully understand how this system-wide behavior solves a problem.

Designing computations for such platforms cannot be done using a conventional programming model. A new computational model that would efficiently utilize the information processing capabilities of these devices is missing. Genetic Algorithms (GA) are an alternative approach for "programming" such devices to perform desired computations. The system learns desired behavior using the principles of evolutionary adaptation. Computation emerges as a collective system behavior while each of the components communicates only with its locally connected neighbors.

The ultimate goal is to develop a programming-language-like environment to design a computation in CA and similar decentralized architectures. One way to do this is to connect the elements of the CA behavior to the parts of a code that cause this behavior. (This approach is similar to making a connection between a genetic disorder and the abnormalities in a genetic makeup that caused the disease.) This approach would identify the building blocks of the CA's behavior. Programming a CA would be equivalent to assembling these blocks in right sequence.

1.2 IDEALIZED MODEL

Systems biology, neuroscience, computational sociology, system ecology, and systems science are examples of fields of studies that analyze the behavior of a system

as whole. Due to physical nature of these systems, research in many of these fields is limited to analysis of an observed phenomenon and the experimentation with a few configuration options. An idealized artificial system is free of temporal, physical, financial, and other constraints, which makes it a potentially useful framework for research purposes. In addition to being easily reconfigurable, an idealized model can serve as a template for physical implementation.

In recent years, the theory and application of cellular automata has experienced a renaissance. CA share characteristics with many physical systems. Examples include many electronic devices, such as field programmable gate arrays, sensor networks, and molecular devices. This makes CA, as a mathematical abstraction of physical systems, well suited for the study of information processing, robustness of computation, and the system's ability to perform computation. The application of CA is especially relevant for nanoscale computing, since proposed nanoscale architectures rely on the same key principles: a large number of identical, simple, and locally connected components that are arranged on a regular two-dimensional lattice [7, 46]. CA are easily reconfigurable to simulate the effect of the environmental artifacts that include various boundary conditions, a noisy environment, a faulty component interconnect, and different update schemas. (Formal definitions are given in Chapter 2.)

1.3 WHAT ARE COMPUTATION AND INFORMATION?

The nature of computation in a system and the definition of what constitutes information in a complex system are concepts that are not well-defined. Short of saying that the meaning of computation is in the eye of the beholder, the definitions of "a computation in a complex system" are wide ranging. For example, Peak et al. suggest that a plant's mechanism of opening and closing its leaves' stomata to maintain optimal water content is similar to cellular automata that perform computational tasks [100]. Many researchers are inspired by the human brain

as the ultimate computer and try to replicate its principles to solve problems in computer vision, signal processing, and other fields of cognitive science [18, 36]. Similarly, what is understood by the word "information" depends on the domain, the scale at which a system is examined, and the investigative perspective on a system. Information in silicon-based devices can be interpreted as a spin of an electron, a presence or a lack of electric charge in a wire, a binary readout from a bus on a circuit board, or a state of a chip's registers.

The meaning of *computation* in the context of a cellular automaton will refer here to the collective behavior of the cells that facilitates the convergence of the lattice towards a task solution, where the task is human-defined. Since such behavior can be interpreted as a "principle" that caused the initial input to converge to the final configuration, this behavior can be interpreted as a general information processing mechanism, or a method of computation. An explanation of such behavior in terms of its function in a lattice would include: a description of the propagation of information-carrying signals through the lattice; a definition of the outcome of interactions among these signals; and a demonstration that the behavior results in a solution for a given task. It should be noted that seemingly chaotic behavior in an automaton does not mean a lack of computation. A lattice might be performing a computation, but the computation's meaning may not be apparent nor understood by a human.

This research focuses on CA with a structured behavior — a pattern of behavior that is clearly visible when we look at the entire lattice. This structured behavior is apparent after the initial chaotic period, when the lattice settles into arranged, organized, almost geometric configurations (resembling rectilinear and curvilinear shapes). A task completion by cooperative global behavior of a lattice is attributed to the assembly and the deformation of these structures over time.

If the collective behavior is synonymous with the mechanism of computation in a system, then the word *information* is analogous to the structures that form in

the lattice. *Information* will refer to the system-wide patterns that change shape and location in space-time, interact with one another, and are modified when they interact with one another. These patterns can be interpreted as the information-carrying structures in the lattice. Finally, the phrase *information processing* simply summarizes the process of formation, transmission, and interaction of these structures in space and time.

The above defined terms for information, information processing, and collective behavior are many times also referred to as an *emergent system behavior*. The meaning of this term goes beyond Aristotle's simple phrase "the whole is more than the sum of its parts". It refers to the occurrence of a structured, system-wide behavior that causes lattice convergence.

An allegorical explanation of these terms can be demonstrated by a popular behavior of soccer fans in a stadium. The *Mexican wave* or *the wave* is a visual effect achieved when a successive group of fans sitting next to each other stand up, raise their hands, and sit down. Each participant rises immediately after their neighbor rose and sits down right after the next neighbor stands up. The result is a wave like pattern that propagates though the stadium. The meaning of the wave is irrelevant; what matters is that it is global scale pattern of behavior that is being transmitted through a crowd. A single spectator who remains standing or sitting down will not disturb the wave; what matters is that a global scale pattern of behavior emerges as a carrier of information (or excitement). This spatio-temporal structure with dynamic properties constitutes information. Now, let's presume that the stadium is an oval with two waves traveling in opposite directions. At some point, the two waves will collide. What happens at the time of collision is analogous to what happens when two information-carrying structures collide in time and space.

1.4 A MODEL OF INFORMATION PROCESSING

There are no automated tools for 2DCA that will answer how collective system behavior emerges in such systems, what are the global behavior patterns that solve a given task, or how to analyze errors if a system fails to find solution to a given problem. Fundamental questions of the nature of information and the mechanisms of information processing lack explanation in such systems.

The first step towards understanding the nature of emergent computation in a two-dimensional cellular automaton is to identify the sites in a lattice with information content. These sites can be distinguished by their function of storing, transferring and modifying information [71, 72, 73]. Statistically-based and regular-language-based filters can be used to identify these sites.

Once the information carrying sites are identified, the next step towards characterizing how CA process information is to build a model that captures the dynamic properties of these sites — also referred to as a *dynamic model*. The sites highlighted in the previous step form system-wide patterns. The dynamic model simplifies these patterns, measures how each pattern changes shape and location over time, and describes the interactions among different information carrying patterns. As soon as the CA lattice settles into organized behavior, the dynamic model is instantiated. From this point on, such a model can, in principle, simulate the behavior of the CA lattice without using the lattice updates—such a model is based purely on abstract "informational" structures.

Up to this point, it is only a hypothesis that a dynamic model can be constructed that accurately describes the mechanisms of computation in a lattice. The assumptions that the structures highlighted as informationally significant carry information and that the dynamic model captures the kinematic properties of these structures must be validated. In order to confirm these hypotheses, the behavior of a dynamic model must be compared with the behavior of the corresponding CA

lattice. If the behavior of the model correctly predicts the information-processing behavior of the CA on a large number of randomly initialized configurations, then we have strong evidence that the model captures the mechanism of information processing in the CA [50, 51, 52, 53].

1.5 DISSERTATION SUMMARY

In this dissertation I investigate the ability of 2DCA as a computational platform. I study this by proposing several problems that require global cooperation among lattice cells to solve given tasks. Subsequently, I attempt to answer how 2DCA solved these tasks. I analyze the complex, emergent, system-wide behavior that was demonstrated by the two-dimensional lattice in order to understand how the lattice computes a solution to a given problem

Chapter 3 introduces several novel tasks for two-dimensional lattices inspired by problems in computer vision, sensor networks, and nano-technology. All the proposed tasks require global cooperation among cells in 2DCA to find solutions. Along with the definition of the tasks, I also illustrate their potential applications, present solutions that show global emergent behavior, and describe the mechanism of computation used by each of the solutions (see Chapter 5 for more details). The results attest to CA's ability to solve various problems using emergent system behavior.

In Chapter 6 I outline the computational mechanics framework to describe the mechanism of collective computation in 2DCA. I describe several approaches for the automatic detection of coherent spatiotemporal structures with information content in Chapter 7. Chapter 8 describes the final step needed to build a model of information processing in 2DCA. I illustrate why the dynamic properties of previously highlighted structures cannot be accurately modeled from the CA's space-time behavior. I present two examples where CA behavior cannot be predicted due to complex (non-linear) lattice behavior.

Chapter 9 will describe related research, and Chapter 10 will summarize the contributions of this work, describe future research directions, and put this work into wider context.

Chapter 2

BACKGROUND

2.1 CELLULAR AUTOMATA

A *cellular automaton* (CA) is a spatially-extended lattice of locally-connected simple processors (cells). CA can be used both to model physical systems and to perform parallel distributed computations.

In a CA, each cell maintains a discrete state and a transition function that maps the cell's current state to its next state. This function is often represented as a lookup table (LUT). The LUT stores all possible configurations of a cell's local neighborhood, which consists of its own current state and the state of its neighboring cells. Each cell updates its state in discrete time steps and the entire lattice is updated synchronously. There are many possible definitions of a *neighborhood*, but here we will define a neighborhood as the cell to be updated along with the cells adjacent to it at a distance of radius r. The number of entries in the LUT will be s^N, where s is the number of possible states and N is the total number of cells in the neighborhood: $(2r+1)^d$ for a square shaped neighborhood with radius r in a d-dimensional lattice, also known as a *Moore neighborhood*. CA typically are given *periodic boundary conditions*, which treat the lattice as a torus.

To transform a cell's state, the value of the cell's state and those of its neighbors are encoded as a lookup index to the LUT that stores a value representing the cell's new state (Figure 2.1 Left) [11, 32, 138]. In this dissertation I focus on homogeneous binary CA, which means that all cells in the CA have the same LUT and each cell has one of two possible states, $s \in \{0, 1\}$. Figure 2.1 shows

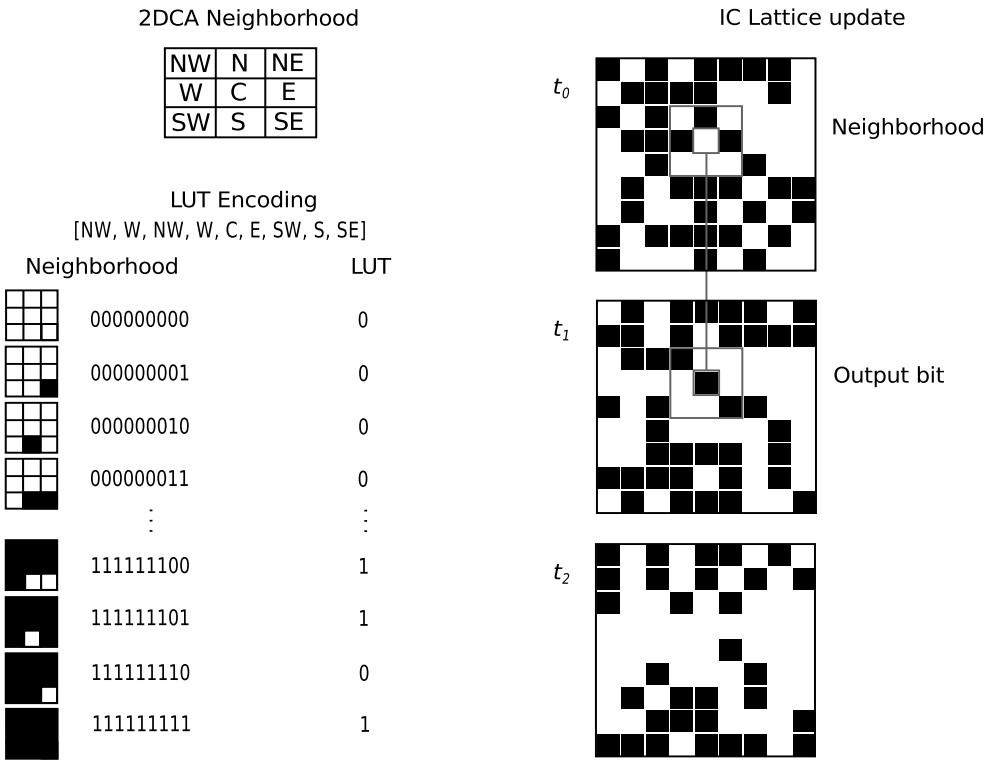


Figure 2.1: **Left top:** A two-dimensional neighborhood of nine cells (radius $r = 1$). **Bottom Left:** A sample look-up table in which all possible neighborhood configurations are listed, along with the update state for the center cell in each neighborhood (Image shows the initial four and the last four neighborhood configurations. The look-up table is shown as a vertical vector on the right.). **Right:** The mechanism of update (using the rule on the left) in a two dimensional binary CA of size 9×9: t_0 is the initial configuration, t_1 is the configuration at next time step, and t_2 is the configuration at the following time step using the rule on the left. A cell in state 0 is colored white while a black cell represents state 1.

the mechanism of updates in a homogeneous two-dimensional binary CA with a neighborhood radius $r = 1$.

CA were invented in the 1940s by Stanislaw Ulam and John von Neumann. Ulam used CA as a mathematical abstraction to study the growth of crystals, while von Neumann used them as an abstraction of a physical system. In order to study the logic of self-reproducing systems, von Neumann introduced the concepts of a cell, state and transition function [11, 19, 131].

Von Neumann's theoretical work on CA had great significance. After the industrial revolution, science was primarily concerned with energy, force and motion, but the concept of self-reproducing systems illustrated how the focus had shifted to information processing, organization, programming, and most importantly, control [11]. The universal computational ability of certain CA was known early on, but harnessing this power continues to intrigue scientists [11, 19, 67, 131, 136].

The best-known example of a CA that supports universal computation is John Conway's Game of Life, introduced in the 1970's [39]. The Game of Life automaton was used as a framework to study emergent behavior and generation of complex patterns in decentralized systems. The ability of CA to generate complex behaviors and patterns attracted not just computer scientists, but game theorists, biologists, physicists, economists, mathematicians and philosophers.

A decade later, Stephen Wolfram conducted an exhaustive study of one-dimensional, binary state, nearest neighbor cellular automata [136, 138, 139]. Such CA were defined by initial input and by rules governing the cell updates. Wolfram defined the simplest of cellular automata as the *elementary cellular automata (ECA)* followed by a number that encodes the rule-table's output bits as a decimal value. For example, rule ECA 109 represents a binary-state cellular automata with neighborhood radius $r = 1$ and the rule's look-up table $0, 1, 1, 0, 1, 1, 0, 1$. The table's bits, in left-to-right ordered, correspond to the output values for the neighborhood configurations from $< 1, 1, 1 >$, $< 1, 1, 0 >$ to $< 0, 0, 1 >$, $< 0, 0, 0 >$. Wolfram proposed the use of one- and two-dimensional cellular automata with neighborhood radius $r = 1$ for the simulation of complex behavior in many fields, including fluid dynamics, materials science, biology, and financial markets.

In order to design CA with desired behavior, the lookup table has to be assigned an output value for each neighborhood configuration. CA rules can be designed by a human or an automated tool. Many attempts have been made to use evolutionary computation techniques to automatically evolve rules for a given task [2, 17, 27, 28,

74, 104, 119, 126]. The tasks described in these works, although simple, require system-wide coordination of information processing. Even though it has been shown that no perfect solution exists for the density classification task, which is described in Section 2.2.1 and 3, it has become a popular benchmark to test how well evolutionary algorithms can design CA rules to perform a desired computation [66].

Since early studies of CA, scientists have observed the ability of CA to exhibit complex behavior, but did not have a way of describing, at a fundamental or algorithmic level, how systems with a lattice-wide cooperative behavior accomplish information processing. A major problem is the difficulty of quantifying computational capability in CAs beyond the general (and not very practical) capability of universal computation. Langton and Packard attempted to correlate a CA's ability for information processing with its generic dynamics [67, 95]. Additional research on information processing in one-dimensional CA followed [27, 42, 43, 51, 89].

Various alternative definitions of CA exist. An architecture worth mentioning is Sipper's non-homogeneous CA with irregular non-local connectivity [119] (See Chapter 9 for more details). In this model, each cell is governed by its own independent set of rules and the connectivity of each cell evolves over time. Sipper's cellular programming algorithm found high performing rules for two-dimensional density classification, synchronization, rectangle image bounding, and image thinning tasks. Even though the algorithm found rather high performing CA rules for given tasks, the non-uniform cell architecture with irregular cell interconnects is likely impractical to fabricate as a computational nano-architecture. Nano-scale devices will likely rely on a massive number of identical components distributed on a regular lattice with components connected only to spatially adjacent neighbors [12, 141].

2.2 COMPUTATION IN CA

Understanding information processing in CA is a first step toward using CA as a practical computational apparatus. In the same manner that fourth and fifth-generation programming languages are used to program von Neumann based machines, the ultimate goal of understanding computation in cellular automata would be the creation of declarative or constraint based programming tools for programming CA.

2.2.1 Early Models

In the early 1970s John Horton Conway published a description of his deceptively simple Game of Life CA [39]. Conway proved that the Game of Life, like von Neumann's self-reproducing automaton, has the power of a universal Turing machine: any program that can be run on a Turing machine can be simulated by the Game of Life with the appropriate initial configuration of states. This initial configuration (IC) encodes both the input and the program to be run on that input. It is interesting that so simple a CA as the Game of Life (as well as even simpler CA—see [139]) has the power of a universal computer. However, the actual application of CA as universal computers is, in general, impractical due to the difficulty of encoding a given program and input as an IC, as well as very long simulation times. Since Conway's work there have been several other demonstrations that certain CA are capable of performing universal computation by either embedding a Turing machine into the CA or by simulating a universal circuit [20, 21, 70, 122].

An alternative use of CA as computers is to design a CA to perform a particular computational task. In such a CA, the initial configuration is the input to the program, the transition function corresponds to the program performing the specific task, and some set of final configurations is interpreted as the output of the computation. The intermediate configurations comprise the actual computation

being done.

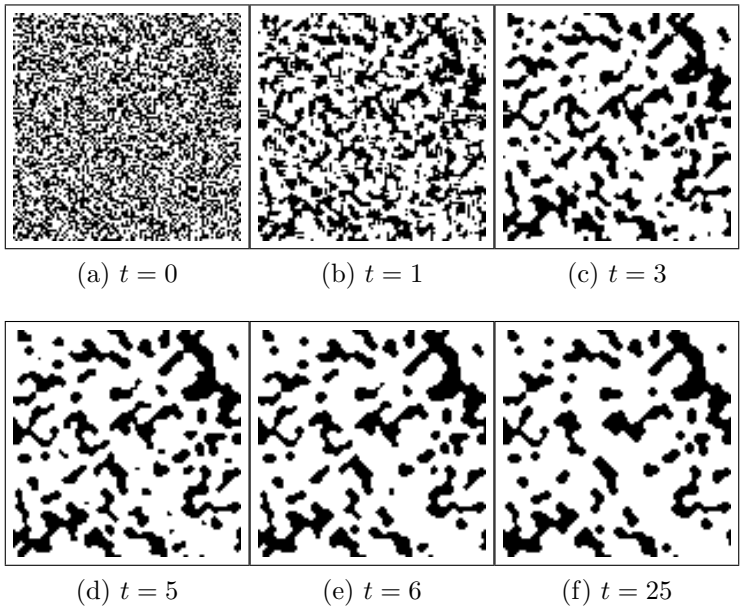
Examples of tasks for which CA have been designed include location management in mobile computing networks [125], classification of initial configuration densities [89], pseudo-random number generation [126], multi-agent synchronization [119], image processing [54], simulation of growth patterns of material microstructures [6], chemical reactions [77], and pedestrian dynamics [112].

The challenge of designing a CA to perform a task includes both the defining of a cell's local neighborhood and boundary conditions, and the construction of a transition function for cells that will produce the desired input-output mapping. Given the CA's state alphabet, neighborhood radius, boundary conditions, and initial configuration, it is the look-up table values that must be set by the "programmer" so that the computation will be performed correctly over all inputs.

In order to study the application of genetic algorithms to designing CA, substantial experimentation has been done using the binary *density classification* (or *majority classification*) task in a one-dimensional lattice. Here, "density" refers to the fraction of 1s in the initial configuration. In this task, a binary-state CA must iterate, after some number of finite steps, to an all-1s configuration if the initial configuration has a majority of cells in state 1, and iterate to an all-0s configuration otherwise. The maximum time allowed for completing this computation is a function of the lattice size.

One "naïve" solution for designing the LUT for this task would be *local majority voting*: set the output bit to 1 for all neighborhood configurations with a majority of 1s, and 0 otherwise. Figure 2.2 gives a time series of lattice configurations illustrating the behavior of this LUT in a two-dimensional binary CA with $N = 99 \times 99$, and $r = 1$, where N denotes the number of cells in the lattice, and r is the neighborhood radius.

Each image is a configuration of 99×99 cells at time steps $t = 0, 1, 3, 5, 7, 25$. The lattice configuration at iteration 25 is at a fixed point.



(a) $t = 0$ (b) $t = 1$ (c) $t = 3$

(d) $t = 5$ (e) $t = 6$ (f) $t = 25$

Figure 2.2: A series of configurations at six time steps, illustrating the behavior of the "naïve" local majority voting rule on a lattice of size $N = 99 \times 99$, and neighborhood radius $r = 1$. The individual cells are colored black for state 1 and white for state 0. The initial configuration is majority white, with density $\rho = 45.41\%$. CA fails to correctly classify the IC. Note that the final lattice configuration is at a fixed point.

In general, this "naïve" CA does not produce the correct global behavior for either class of initial configurations: a final all-1s configuration when the initial density of 1s is greater than 50% or a final all 0s for configurations with initial density less than 50%. This illustrates the general trend that human intuition often fails when trying to capture emergent collective behavior by manipulating individual bits in the lookup table.

Packard [95] was the first to use genetic algorithms to evolve CA look-up tables to perform the density classification task. Langton had shown that generic CA behavior seemed to undergo a sequence of phase transitions—from simple to "complex" to chaotic as the fraction of the lookup table ones is increased from 0 to 0.5 [68]. Correlations between "complex" regions and computational capability

in CAs have been hinted at in further work, but have not been definitively established. A new approach to analyze computation in one and two-dimensional CA was needed: a model that would identify the computationally relevant structures in a CA's space-time diagram and identify sites of information storage, transfer, and modification.

2.2.2 Computation in 1DCA via "Particles"

In order to analyze computation in 1DCA, the information-carrying sites must be identified first, then a model describing the dynamics of information transfer has to be developed, and finally the analysis of information modification completes the view of computation in the CA.

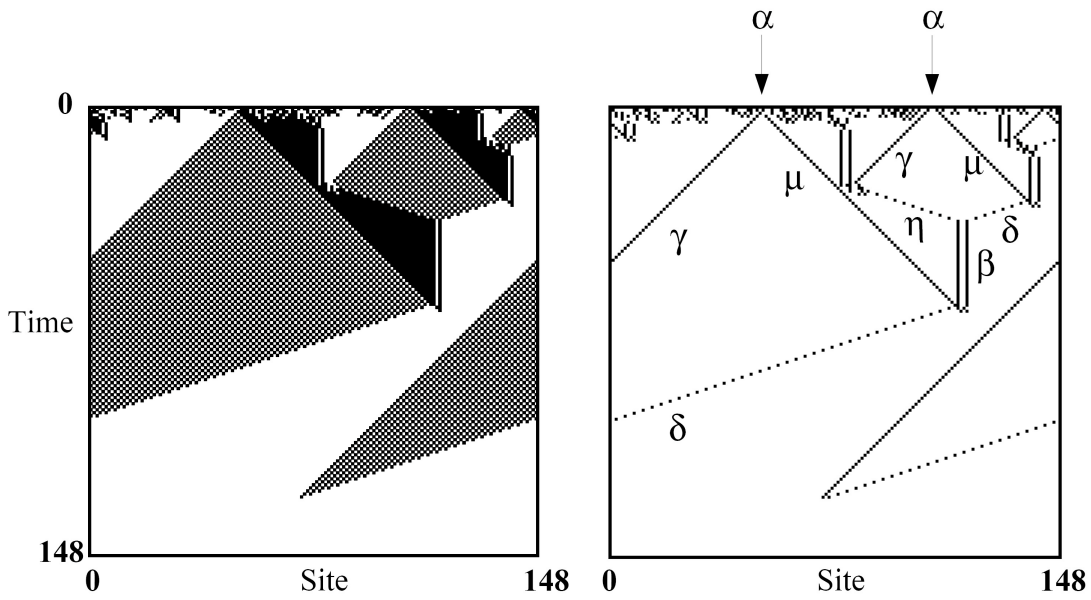


Figure 2.3: Analysis of GA-evolved CA for the density classification task. **Left:** A diagram of the CA's behavior on a 149 cell lattice over 148 time steps, starting from a random IC. The regular domains consist of all white, all black, or checkerboard regions. **Right:** Space-time diagram after regular domains are filtered out. Reprinted from (Crutchfield, Mitchell, & Das, 2003).

Figure 2.3 displays typical space-time behavior of a one-dimensional binary-state CA with toroidal boundary conditions that was evolved by a GA to perform

the density classification task. In the left diagram, the CA is shown starting from a random IC with the one-dimensional lattice displayed horizontally and with time going down the page. In the first few time-steps the CA creates uniform regions of black, white, and checkerboard patterns. The cells inside of these regions only store information; in other words, no change of information needs to be recorded. The information storing sites retain the same information, and a space-time diagram shows these sites as a regular pattern. The sites that unexpectedly change their state were prompted to do so by a signal. The sites that carry information that causes a change of stored state are called *information transfer sites*. In order to uncover the information transfer sites in these diagrams, we filter out the information storing cells; the remaining cells represent the sites involved in the transfer of information in the lattice [42, 43]. The identification of such cells is the key to understanding information processing in the CA.

When the sites at which information is transfered are identified at each time step, the spatial pattern of these sites represents the dynamic properties of information propagating through the lattice. Using earlier work by Hanson and Crutchfield on characterizing computation in CA [42, 43], Das, Mitchell and Crutchfield gave an information-processing interpretation of the dynamics exhibited by the evolved CA in terms of *regular domains* and *particles* [43]. This work was extended by Das, Crutchfield, Mitchell, and Hanson [27] and Hordijk, Crutchfield and Mitchell [51]. In particular these groups showed that when regular domains—patterns described by simple regular languages—are filtered out of 1DCA space-time behavior, the boundaries between these domains become apparent and can be interpreted as information-carrying "particles." These particles can characterize the computation carried out by a particular CA [28, 43].

Hordijk et al. developed quantitative models to analyze the emergent behavior in CA. The models captured only the kinematics of particle movement and interactions. These models accurately predicted the computational performance

of evolved CA [50, 51]. The models explained computation in CA by linking the notion of the emergent global behavior to a model, and the model to a performance measure. Figure 2.4 compares the accuracy of CA rule performance (white bars) and the performance predicted by the model (black bars) for a CA rule evolved for one-dimensional density classification task. The plot shows five rules that represent typical behavior for each epoch of improvement observed during the GA execution. Since the CA's performance (white bars) corresponds to the model's performance (black bars), these results support Hordijk et al.'s hypothesis that the particle–level model of the CA's dynamic structures correctly captures the mechanism of the CA's emergent computation necessary to perform a given task [50].

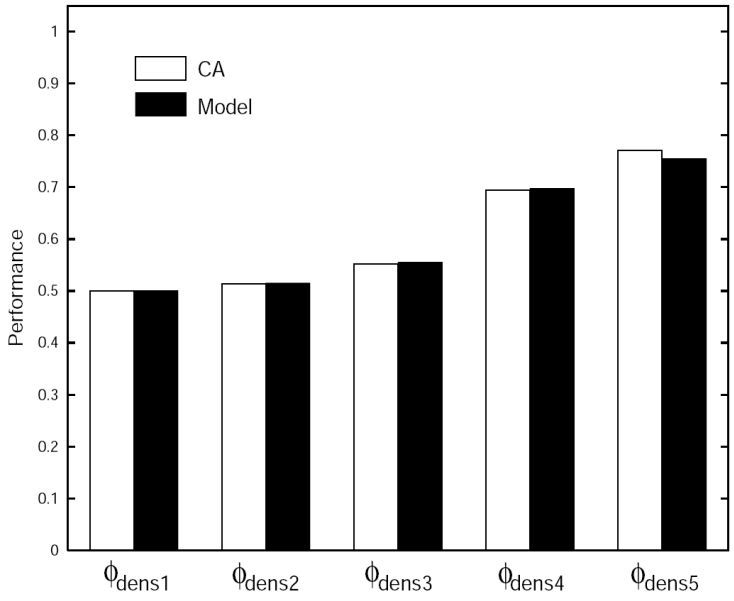


Figure 2.4: Comparison of CA rule performance (white bars) and the performance predicted by a quantitative model (black bars) for a typical CA rule evolved for 1D density classification task. Five CA rules (ϕ_{dens1} to ϕ_{dens5}) are representative of epochs of rule improvement during GA run – stages of significant improvement in the rule's fitness. Reprinted from Wim Hordijk's PhD thesis [50].

Land and Belew [66] proved that no two-state CA lattice with identically defined cells distributed on a regular grid can perform the density classification task perfectly. However, the maximum possible performance for CA on this task is not

known.

The density classification task remains a popular benchmark for studying the evolution of CA with GAs, since the task requires collective behavior: the decision about the global density of the IC is based on information only from each local neighborhood. A second benchmark task is *global synchronization*. This task requires a CA with any initial configuration to synchronize all of its cells to the same state (all 1s or 0s) while in the next time step all cells will change state to the opposite one. Again, this behavior requires global coordination based on local communication. Das et al. showed that an analysis in terms of particles and their interactions was also possible for this task [28, 103].

2.2.3 Computation in 2DCA

A lot of work has been done to analyze the mechanism of information processing in 1DCA, but the principles of computation in 2DCA have received little attention [73, 114]. The lack of interest might be due to the following reasons. First, very few problems are defined and solved by emergent system behavior in two dimensions. Tasks for decentralized systems, such as for mobile computing and sensor networks, might benefit from emergent collective computation. Instead, well-understood network algorithms, such as graph search algorithms by Dijkstra, Tarjan, Kruskal, and Prim [22], are used to solve many network problems. Second, there are no suitable tools to analyze the emergent behavior in two dimensions. The analytical tools lack both the filtering methods to identify coherent spatio-temporal patterns with information content and the kinematic model to accurately describe the evolution of these patterns over time.

This dissertation addresses all three problems by proposing novel tasks that require system-wide cooperation, extending statistically based filtering methods to highlight informationally significant sites, and laying groundwork for a dynamic model to describe the mechanism of information processing in 2DCA (see Chapters

3, 7, and 8 respectively).

Chapter 3

TASKS

This dissertation explores a general hypothesis about 2DCA: that these structures are capable of solving non-trivial computational tasks that require system-wide cooperative behavior. The *density classification* and *global synchronization* tasks are two well-known problems used by researchers to test the ability of two-state CA to perform global computation. In addition to these tasks, this chapter introduces *spatial density niching* and *rectangular image bounding* tasks to gauge the ability of CA to perform information processing in two-dimensional cellular automata. Since the last two tasks were inspired by potential image processing applications, I will refer to these tasks as the *image processing tasks*. Even though the global synchronization task does not perform classification, the task solutions have a "binary" meaning of correctness. I will loosely use the term *classification tasks* to refer to the density classification and the global synchronization tasks.

In this chapter, each task is described in terms of input and desired output that represents the ideal CA configuration without noise. Along with the task definitions, I also outline potential applications of these tasks.

3.1 DESIRED ATTRIBUTES OF TASKS

First, let's look at what makes the tasks proposed in this chapter so difficult. Figure 3.1 illustrates a sample problem in an initial configuration presented to a 2DCA. The CA's task is to locate the dark rectangle in the middle of the image (see Section 3.4 for more details). A human observer can immediately tell that the

image has an area with higher and lower pixel densities, and that the higher density region in the center of the image forms a darker color rectangle. This is because the image is viewed from a distance. It is evaluated by its overall appearance – a global scope.

The very definition of a CA as a distributed and decentralized network of simple, locally connected processors contradicts the notion that a CA has a built-in mechanism capable of making observations about a global state of the lattice (or a feature larger than the neighborhood radius). No single cell or small collection of cells has the global information that the sample initial configuration contains areas with different densities nor that the dark block has a shape of a rectangle. The lack of knowledge at the local level about the attributes of the initial configuration is what makes the problem so difficult for a CA to solve. All that a CA "sees", at each site of the lattice, is a configuration of the local neighborhood. This concept is illustrated by the neighborhood configurations along the image sides (Figure 3.1). Even for a human with the knowledge of the task, it is difficult to decide if these patches belong to the inside or the outside of the darker rectangle, or if the neighborhood configuration forms the rectangle's edge.

Although CA have successfully solved numerous problems [6, 54, 77, 89, 119, 125, 126], not all tasks require global collective behavior. Tasks such as image thinning [108] and pixel filling [102] need only information about local neighborhood configurations.

This research focuses on tasks that require collective system behavior. To achieve this requirement, the features to recognize or to reason about must be larger than the neighborhood radius.

3.2 DENSITY CLASSIFICATION

The definition of the *density classification* task, also known as a *majority classification*, is to decide if the fraction of 1s in the initial configurations is greater than

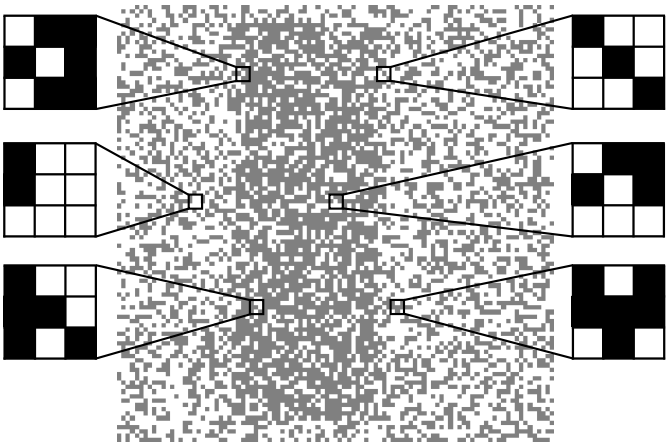


Figure 3.1: An example of a task to detect the darker rectangle in the center of the image. The neighborhood configurations randomly selected from the image are showed on either side of the image. Deciding if these configurations belong to the inside, outside, or the border of the rectangle is difficult without looking at the whole image.

the 50% threshold. The lattice should converge into an all black configuration if the lattice started with the majority of cells in state 1 (black), and settle into all white if the initial configuration had density less than 50%. An initial configuration is misclassified if a lattice converges to a wrong configuration, or after a fixed number of updates a lattice has both black and white regions. (The number of CA updates is a parameter of the lattice size).

This task has a wide range of potential applications. Well-suited problems for this task include decision making about an environment where the occurrence of observed events cannot exceed a predefined threshold. Examples include deciding air safety if the density of airborne particulates exceed an allowed threshold and emergent detection of an earthquake from an array of noisy vibration sensitive sensors.

The hardest instances of the density classification problem are the initial configurations with the density close to 50%. Although the task definition can be modified to classify the lattice with other threshold densities (such as 20%, 30%,

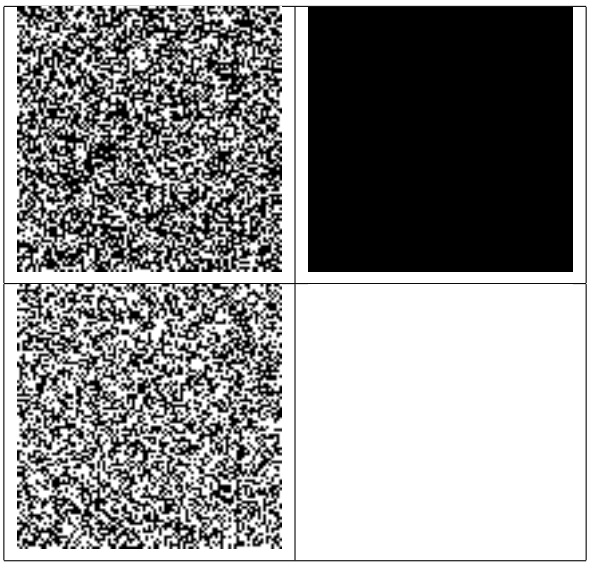


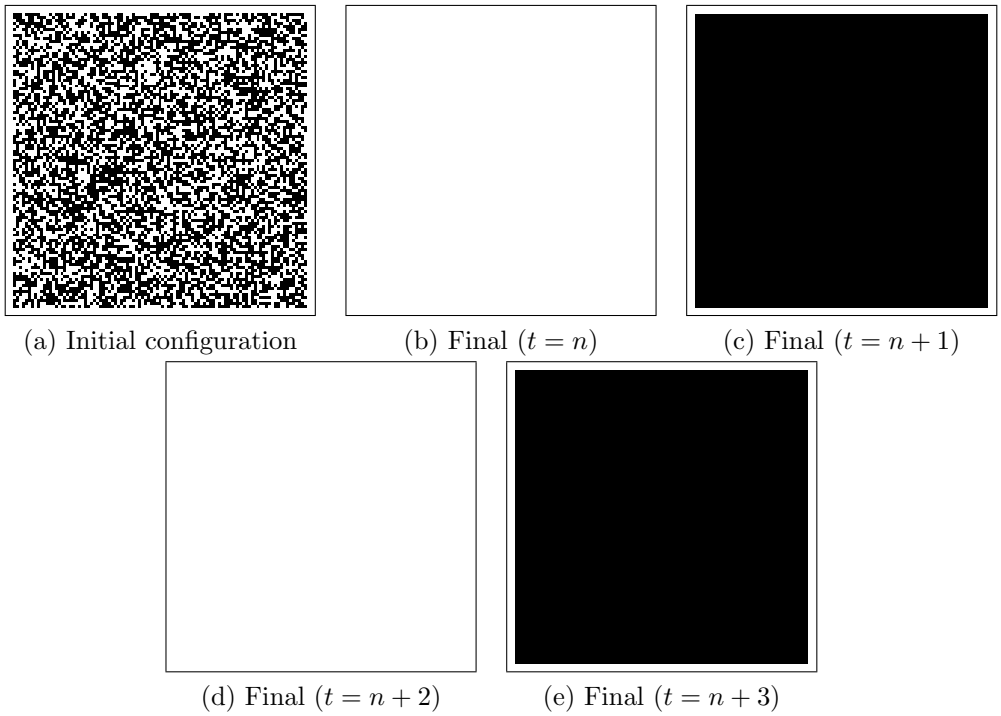
Figure 3.2: Example of a Density Classification task. The top row illustrates the initial configuration with a majority of cells in state 1 (density 54.1%) where the bottom row shows the initial configuration with 47.4% density. The columns on the right represent the correct solutions to the initial configurations displayed in the left column.

70%, or 90%) these tasks are not any easier if the IC density is close to the threshold value. GA-evolved rules must have sophisticated methods of communicating information through the lattice to decide which density dominates the lattice on a global scale.

3.3 GLOBAL SYNCHRONIZATION

Imagine that each cell of the lattice represents a simple processor. The processor is On (active) if its state is 1, and the processor is Off (idle) if its state 0. The *global Synchronization* task requires coordination of all processors to arrive in the same state. After such agreement is reached, in the next update all processors change their state to the opposite one. These two lattice configurations will alternate indefinitely. Figure 3.3 shows the task's input as a randomly initialized lattice that after a series of updates converges to an alternating configurations of all 1s

and all $0s$ (configurations of all white and all black).



(a) Initial configuration (b) Final ($t = n$) (c) Final ($t = n + 1$)

(d) Final ($t = n + 2$) (e) Final ($t = n + 3$)

Figure 3.3: Example of a Global Synchronization task. (a) Sample IC representing the initial state of the processors, while (b) - (e) show the desired output as an alternation of all processors On (all white) and Off (all black) configuration.

One of the primary design principles of a modern computer architecture is its central clock. This simple synchronization mechanism provides common phase for communication among the computer's components, which is the fundamental requirement for its operation. A common clock design has been used in the following fields: distributed computing, sensor networks, and parallel computing, to name a few. In some applications, providing a centralized common clock synchronization might not be feasible due to location (devices are too far apart for synchronization broadcast), device connectivity (not all devices are directly connected to the central controller), asynchronous design (some devices might idle or sleep during broadcast), and high clock frequency (devices located far from the central clock will experience clock skew).

In the absence of a central controller, global synchronization of spatially distributed processors is equally difficult. Although the interactions among processors within the neighborhood radius can provide regions of local synchrony, spatially distant regions might settle into regions with the opposite state. In order to achieve global synchronization, the differences between adjacent regions must be resolved. Such a mechanism requires information communication on a global level.

Applications of global synchronization apply to any distributed architecture that requires data-consistency during read and write operations and correctness of algorithms designed to provide network functionality [31]. Networks utilizing global synchronization include sensor networks to detect earthquakes despite varying sensor proximity, multi-core computer architectures to exchange information between individual cores despite high core frequency causing clock-skew, and power-aware controllers that schedule devices to turn off and on to save energy despite the network's asynchronous design.

3.4 SPATIAL DENSITY NICHING

The definition of this task is to identify a non-uniform density locale (a niche) in an otherwise uniformly distributed random IC. Three versions of the Spatial Density Niching task are (1) positive: one or more local areas have greater density than the rest of the lattice (Figure 3.4 Top) (2) negative: the image contains areas with lower density than the rest of the lattice (Figure 3.4 Bottom) and (3) mixed: the lattice contains both negative and positive local niches.

These problems require a CA to "highlight" the niche regions with distribution higher (or lower) than the rest of the lattice. If the foreground artifact has higher density than the background distribution, the desired solution should turn the foreground niche all black and turn the background all white (Figure 3.4 Top). Figure 3.4 Bottom shows the opposite setup, where the low density foreground turns all white while the high density background turns all black. The CA is not

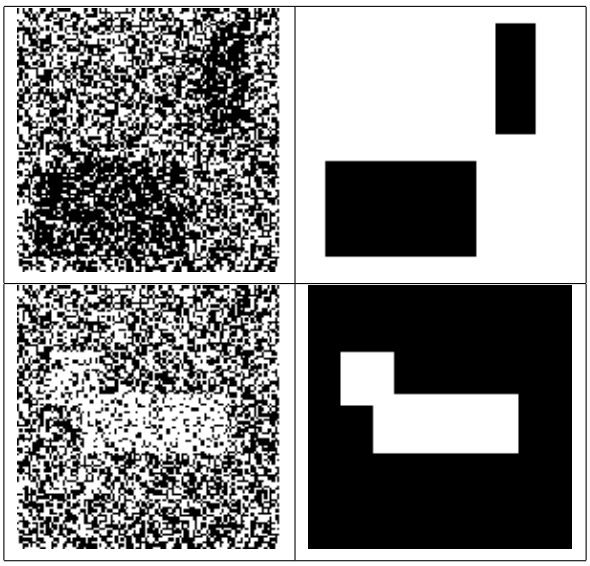


Figure 3.4: Example of a Spatial Density Niche task. The top row illustrates the positive task variant and the bottom row has the negative task variant. The left column represents random ICs and the right column has the desired task solution. **Top Left:** The configuration contains areas with higher density than the surrounding lattice.**Top Right:** The higher density areas turn black and the rest of the lattice is white. **Bottom Left:** The configuration contains rectangles with lower density **Bottom Right:** The low density areas turn white and the rest of the lattice is black.

given either the background or the foreground distribution values, and it has to recognize the areas with different densities via the collective actions of the cells.

The change of local density in real-life applications might represent different feature characteristics such as defects in material, abnormal tissue growth in biomedical images, hairline fractures in tooth enamel visible in X-Rays, and holes in sensor networks that represent obstructions or non-functional devices. Examples of systems for solving local density niching tasks include engineered nano-architectures for metallurgical detection of crystal structures in alloys and automated detection of material fatigue and fractures.

3.5 RECTANGLE IMAGE BOUNDING

This task will box all outlying pixels into a black rectangle (image pixels are represented by CA cells). Two variants of this task are: (1) thick image bounding, where areas with high pixel density are turned into a black rectangle and (2) sparse image bounding, where only a few pixels appear on the image but the goal is the same. Figure 3.5 illustrates these tasks with sample initial configuration on the left and the task solution on the right.

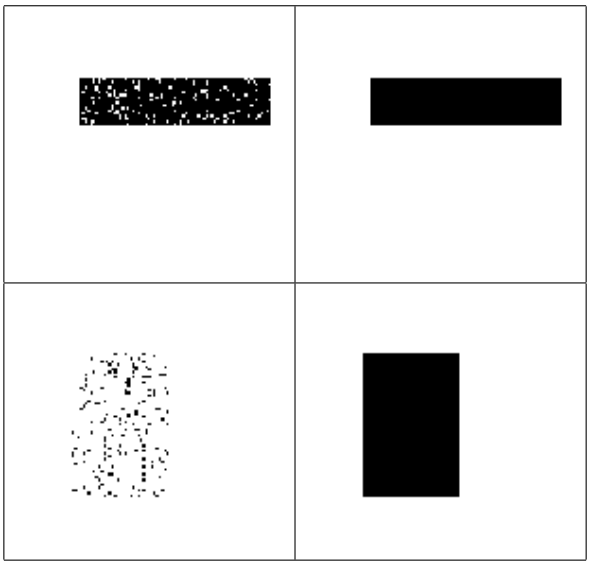


Figure 3.5: Example of a Rectangle Image Bounding task. The top row illustrates thick image bounding and the lower two images illustrate the sparse image bounding tasks. **Left:** Sample ICs with the images to be bound. **Right:** The solutions to the task. All image pixels are bound by an outlying rectangle.

The difficulty of this task is similar to that of the spatial density niching task. Let's consider an IC with only four black pixels and any two black pixels are at least ten cells apart in both the horizontal and vertical directions. Each black pixel marks the location of the bounding box edge, which outlines a rectangle several times greater than a cell's neighborhood. Although these perimeter pixels are outside of most cells' neighborhoods, the CA rule has to form precise lines for the sides of the rectangle, define corners where the sides meet, and decide if the

cell is inside or outside of the bounding box. To successfully accomplish this task, a CA's global behavior has to communicate the location of these bounding pixels through the lattice.

The only difference between the dense and the sparse image bounding variants is their difficulty. The dense image bounding is simply filling in the rectangle's white pixels. The decision to fill in the pixel or not can be easily inferred from the neighborhood density. The sparse image bounding is much harder, as described in the example above, because information about the location of the bounding rectangle is outside of the cells' neighborhood.

Examples of applications of this task might include: start cooling a circuit board if the bounding box area of heat triggered sensors exceeds a threshold surface value, locate and repair corrupted image pixels inside of a bounding box, bound pixels making up a star into a rectangle for image tracking in astronomy, and create a bounding box around an image feature such as typed characters for optical character recognition or the license plates in a given image.

3.6 SUMMARY

The tasks proposed in this chapter have simple definitions. In order to evolve high-accuracy solutions for real-world problems, the tasks and the structure of the CA must be defined as accurately as possible to represent the actual application the solution is built for. Some tasks might require only approximate solutions, in which case rules with lower accuracy are acceptable. Finding solutions with high accuracy might require using domain-specific search approaches. Different approaches have been used to find solutions for problems in 1D and 2D [2, 35, 94, 120, 140]. Further investigation should follow to explore which technique(s) yield the best-quality solutions for the above described problems and why. (See Chapter 9 for a summary of several related topics.)

Although each task definition is accompanied by a list of potential applications,

a CA solution for a real-world problem is yet to be found. The task definitions and the results presented in Chapter 5 attest to the versatility and power of a CA with synchronous updates, wraparound boundaries, regular component interconnect, and fully functional cells. To account for the defects during manufacturing and failures while operating a potential CA-like device, the original CA definition has to be relaxed. Different boundary conditions, asynchronous timing of updates, faulty components, and imperfect component connectivity are a few alternatives for CA-like architectures. A detailed investigation has to be conducted on how the changes in the CA architecture affects its ability to solve computational problems.

Chapter 4

EVOLVING CELLULAR AUTOMATA WITH GENETIC ALGORITHMS

Most research using GAs to evolve CA to perform complex global tasks has focused on one-dimensional CA. Jimènez-Morales, Crutchfield, and Mitchell [55] performed a preliminary study of extending the density classification task from one dimension to two dimensions. Unfortunately, they did not extend the analysis of information processing from 1DCA to 2DCA. This chapter describes several evolutionary techniques that I used to evolve rules for the proposed tasks.

4.1 RULE PERFORMANCE AND FITNESS

In order to assess the accuracy of the GA-evolved solutions, each of the rules is executed on 10^4 randomly initialized starting configurations (*test cases*). The *rule performance* of the density classification and global synchronization tasks is the fraction of correctly classified test cases on 10^4 randomly generated initial configurations. If a lattice converges to the correct configuration, the classification is correct, otherwise no partial credit is awarded and the classification is incorrect. The *rule performance* for spatial density niching and rectangular image bounding is defined as the average fitness value of the lattice at its final configuration over all 10^4 randomly initialized starting configurations.

The density classification and global synchronization tasks have a binary definition of a rule's fitness, while the spatial density niching and the rectangular image bounding tasks use a decimal value to express this concept. For these latter two tasks, the *fitness* of a lattice is the inverse of a weighted sum of all misclassified

pixels with respect to the ideal output. The weight of a misclassified pixel is its shortest distance from the edge of a feature. For example, if the idealized output is a black rectangle on a white background, then take all the white pixels that are located inside of the ideal black rectangle, calculate the distance of each such pixel from the ideal rectangle's closest edge, and add up all these distances. Next, repeat this process for all black pixels that appear on what ideally should be the white background; here, the distances to be summed are the distances from these black pixels to the ideal rectangle's closest edge. The lattice's fitness is an inverse of the sum of all distances for all such pixels with incorrect final state.

4.2 EVOLVING CELLULAR AUTOMATA WITH GENETIC ALGORITHMS

Genetic Algorithms (GAs) are a group of stochastic search algorithms, inspired by the Darwinian model of evolution, that have been proven successful for solving various difficult problems [3, 4, 86].

A GA works as follows: (1) A population of individuals ("chromosomes") representing candidate solutions to a given problem is initially generated at random. (2) The fitness of each individual is calculated as a function of its quality as a solution. (3) The fittest individuals are then selected to be the parents of a new generation of candidate solutions. Offspring are created from parents via copying, random mutation, and crossover. Once a new generation of individuals is created, the process returns to step two. This entire process is iterated for some number of generations with a goal of creating one or more highly fit individuals that are good solutions to the given problem.

GAs have been used by a number of groups to evolve LUTs for binary CA [2, 17, 27, 28, 74, 104, 119, 126]. The individuals in the GA population are LUTs, typically encoded as binary strings. Figure 4.1 shows a mechanism of encoding LUTs as binary strings. For example, the decimal value for the neighborhood

111111011 is 507. The value stored in the look-up table's 507th position represents the new value for the neighborhood's center cell. Using the sample LUT from Figure 4.1, the CA would then update the value of the neighborhood's center cell 111**1**1011 to 0.

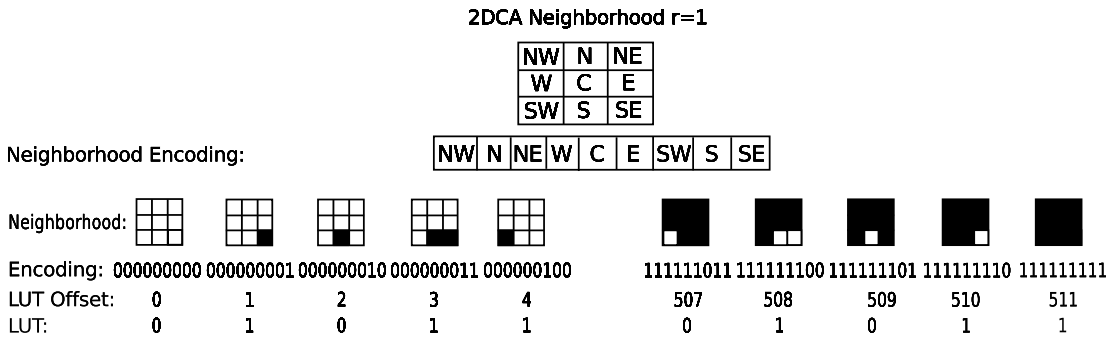


Figure 4.1: Lookup table encoding for 2D CA with neighborhood $r = 1$. All permutations of neighborhood values are encoded as an offset to the LUT. The LUT bit represents a new value for the center cell of the neighborhood. The binary string (LUT) encodes an individual's chromosome used by evolution. The length of the binary string encoding of the LUT is $2^{(2r+1)^d}$.

The fitness of a LUT is a measure of how well the corresponding CA performs a given task after a fixed number of time steps, starting from a number of *training* initial configurations. For example, given the density classification task, the fitness of a LUT is calculated by running the corresponding CA on some number k of random initial configurations, and returning the fraction of those k on which the CA produces the correct final configuration (all 1s for initial configurations with majority 1s, all 0s otherwise). The set of random training ICs is typically regenerated at each generation.

For LUTs represented as bit strings, crossover is applied to two parents by randomly selecting a crossover point, so that each child inherits one segment of bits from each parent. Next, each child is subject to a mutation, where the genome's individual bits are subject to a bit complement with a low probability. An example of the reproduction process is illustrated in Figure 4.2 for sample binary strings of 8 bits long. Here, one of two children is chosen for survival at random and placed

in an offspring population. This process is repeated until the offspring population is filled. Before a new evolutionary cycle begins, the newly created population of offspring replaces the previous population of parents.

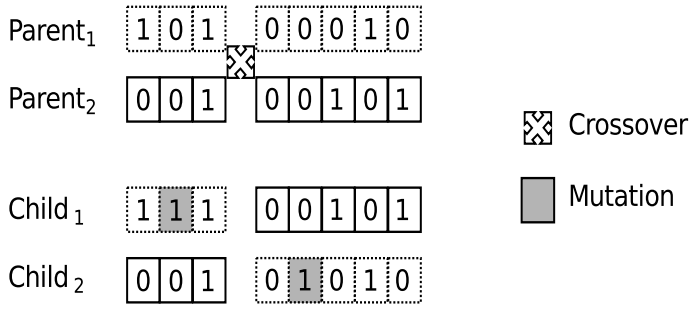


Figure 4.2: Reproduction applied to $Parent_1$ and $Parent_2$ producing $Child_1$ and $Child_2$. The one-point crossover is performed at a randomly selected crossover point (bit 3) and a mutation is performed on bits 2 and 5 in $Child_1$ and $Child_2$ respectively.

4.2.1 Coevolution

Coevolution is an extension of the GA which was inspired by host-parasite coevolution in nature and first introduced by Hillis [48]. The main idea, in the context of evolving CA, is that randomly generated training ICs will not continually challenge evolving candidate solutions; this lack of continual challenge can cause evolution to stagnate. Coevolution addresses this problem by evolving two populations simultaneously—candidate solutions and training ICs—also referred to respectively as *hosts* and *parasites*. The hosts obtain high fitness by performing well on many of the parasites, whereas the parasites obtain high fitness by being challenging to the hosts. Coevolving both populations engages hosts and parasites in a mutual competition to achieve increasingly better results [10, 37, 134].

Successful applications of coevolutionary learning include discovery of minimal sorting networks, training artificial neural networks for robotics, function induction from data, and evolving game strategies [13, 48, 96, 107, 134, 135]. Coevolution

also improved upon GA results on evolving CA rules for density classification [56].

In the context of evolving CA, the LUT candidate solutions are hosts, and the ICs are parasites. The fitness of a host is the fraction of correctly evaluated ICs from the parasite population. The fitness of a parasite is a function of the number of hosts that failed to correctly classify it.

4.2.2 Spatially Extended GA

Pagie et al. [96, 97, 98] and Mitchell et al. [90, 135], among others, have found that embedding the host and parasite populations into a spatial grid, in which hosts and parasites compete and evolve locally, significantly improves the performance of coevolution on evolving CA.

This extension of coevolutionary learning distributes populations into a two dimensional grid where each location contains one host and one parasite. The algorithm requires a much smaller number of fitness evaluations per evolutionary run, since the fitness of each individual is evaluated against individuals only in its local neighborhood. The results show that spatially confined reproduction and selection helps to sustain a higher genetic diversity of evolving populations. At this time, the observed results are considered circumstantial evidence that evolving spatially extended populations leads to high efficiency of finding high quality solutions, as the exact role of space in coevolutionary learning is still unknown.

4.3 GENETIC ALGORITHMS USED IN THIS WORK

All genetic algorithms represented the rule's lookup table as a binary array of 512 bits for a binary state CA where each cell is connected to the spatially adjacent cells in a 3×3 Moore neighborhood. Each genome was randomly initialized at the beginning of each execution of the genetic algorithm using a 0.5 binomial distribution. This discrete probability distribution of genomes' densities was achieved by

setting each bit in each genome to 0 if a randomly generated value from interval $< 0, 1 >$ for given bit was less than 0.5 and set to 1 otherwise.

The test cases for the density classification and global synchronization tasks were represented by 361-bit arrays which corresponds to a 19×19 cell lattice. The initial population of the training cases was randomly initialized with the uniform distribution. First, a random density value was generated for each genome from the interval $< 0, 1 >$, then each bit in a given genome was initialized to 0 if a randomly generated value for a given bit (from interval $< 0, 1 >$) was less than genome's density, otherwise the given bit was set to 1. Evaluating the fitness of a candidate solution on a training CA lattice was performed by updating the CA lattice until it converged to a desired configuration or for no more than $M = 300$ iterations.

The training and tests used for the image processing tasks were represented by 2401-bit arrays (49×49 cell lattices). A considerably larger lattice size compared with the classification tasks was used to better distinguish the foreground feature(s) from the background. The initial training cases were initialized randomly using a uniform distribution with the foreground and background densities varying by at least 15%. The initialization was very similar to the process of instantiating the population of training cases for the classification tasks, with the exception of generating two random densities (corresponding to the foreground and the background) that vary by at least 15%. Each bit was initialized randomly with a uniform distribution. The process of initialization is the same as described in the previous paragraph: a new random value is generated for each bit in the interval $< 0, 1 >$; a bit is set to 0, if its random value is less than the density associated with its location (foreground vs. background), otherwise the bit is set to 1. Due to the larger lattice size, the lattice was updated for $M = 900$ time steps. The weighted sum of misclassified pixels was calculated from the lattice's final time-step configuration.

All genetic algorithms used populations of 400 candidate solutions and 400 training cases. The algorithms were executed for a maximum of 4000 evolutionary steps, unless a high quality solution was discovered. Such solutions correspond to the density classification and the global synchronization rules with the respective performances of 70% and 90% or better. The early stopping criteria for the spatial density niching and rectangular image bounding rules was performance of 0.002 or better (calculated as an inverse of 5% of misclassified lattice pixels with 5 pixels average distance from the ideal rectangle's closest edge).

One-point crossover followed by a bitwise mutation were used as the reproduction operators for evolving candidate solutions in all algorithms. The crossover point in the parents' genome was selected at random from the interval $< 0, 512 >$ with uniform distribution. Each bit of the resulting offspring arrays was subject to mutation with probability 2×10^{-3}, which corresponds to one mutation per genome. One of the resulting offspring was selected at random and placed into the offspring population. Below I describe four versions of the GA used in this research: the Standard GA, Non-spatial Coevolution, Spatial Evolution, and Spatial Coevolution.

4.3.1 Standard GA

In the standard GA, the fitness of each candidate solution was assessed using 20 randomly selected training cases with replacement. The training cases were re-sampled for each candidate solution, and all training cases were re-generated at the end of each generation.

Tournament selection was used to select two parent genomes for reproduction. The tournament group size of 9 was populated by randomly selected genomes without bias from the population of candidate solutions. First, the tournament group was sorted in descending order of individual's fitness. Next, individual genomes were assigned ranks in range 1–9 in order of fitness. Finally, the first parent for

reproduction was selected from the group at random with bias of $0.5^{Individual's\ rank}$. (Note: the last two ranked individuals had the same bias to make the sum of nine bias values equal to 1.) For generating offspring by one-point crossover, the second parent was selected from the tournament group at random without bias. (See [98] for more details)

4.3.2 Non-Spatial Coevolution

Non-spatial coevolution used the same assessment of fitness for each candidate solution as in the standard GA. The training cases were not regenerated at the end of each generation. Instead, at each generation a population of offspring training cases was created by evolving the densities of each training case.

The classification task training cases were first assigned a fitness value. Each training case had a fitness calculated as a normalized value of the total number of candidate solutions that misclassified this particular training case (number of candidate solution defeats). A cumulative count was kept for each training case during the assessment of the fitness of the entire population of candidate solutions. Next, an offspring was generated by varying the parent density by a random value from the interval $< -0.1, 0.1 >$. The offspring training case was regenerated with a new density.

For the image processing tasks, a training case's fitness value is calculated similarly to the classification tasks. Instead of a cumulative count of candidate defeats, the sum is calculated using the fitness values of a lattice at the final configuration. Since there is no notion of misclassification, the fitness values are used from all candidate solutions that used this particular training case for fitness evaluation. An offspring is generated the same way as in evolving solutions for the classification tasks. The reproduction operators perform a variation of one of the densities – either a foreground feature's or the background's density.

Tournament selection was used (as in the Standard GA) to select genomes for

reproduction during evolution of candidate solution and training case populations.

4.3.3 Spatial Evolution

The spatial evolution algorithm arranged the populations of candidate solutions and training cases on a 20×20 grid. Each grid site contained one candidate solution and one training case. The only changes to the algorithms were done when assigning fitness values and selecting genomes for reproduction. Instead of using randomly selected genomes from the entire population, the genome's fitness was assessed against 9 training cases from a local neighborhood. Generating an offspring was done at each grid site. When evolving the population of candidate solutions, each genome was replaced by an offspring. Instead of randomly selecting individuals for reproduction from the entire population, the tournament group consisted of genomes from a local neighborhood.

4.3.4 Spatial Coevolution

Spatial co-evolution works the same as spatial evolution with the exception that the population of training cases is evolved at the same time as the population of the candidate solutions. The fitness of a candidate solution is assessed with respect to the spatially adjacent training cases; and vice-versa, the fitness of a training case is evaluated with respect to the adjacent candidate solutions. The fitness value of a test case is the number of its "unsuccessful" evaluations by the spatially adjacent candidate solutions. The density classification and the global synchronization define an unsuccessful evaluation as a lattice configuration other than the tasks' ideal output. For the image processing tasks an unsuccessful evaluation is when the fitness of a lattice configuration at the final time-step is above a threshold value of 600. This value corresponds to 5% of misclassified pixels with 5 pixels minimum average-distance from the ideal rectangle's closes edge.

Similarly to the spatial evolution, the reproduction was executed at each population grid-site. The parents for reproduction were chosen by the tournament selection that used only genomes from the local neighborhood. Each genome in the parent populations was replaced by an offspring that was generated the same way (for both populations of candidate solutions and training cases) as in the non-spatial coevolution with the additional constraint of populations' spatial distribution.

Chapter 5

RESULTS OF EVOLVING CELLULAR AUTOMATA WITH GENETIC ALGORITHMS

Here I present the GA-evolved rules for the tasks defined in Chapter 3. The rules are described in terms of the behavior that appears to be the mechanism used to solve a given problem.

Unless otherwise stated, the solutions to the proposed problems were found using the standard genetic algorithm. Although evolution, coevolution, and spatial evolution and coevolution were used to find solutions for the tasks, the standard GA outperformed the rest of the algorithms, and found the rules with the highest performance. The discussion of the best-performing results focuses only on the rules with the highest performance found by the standard GA.

5.1 THE BEST GA EVOLVED RULES

The standard GA evolved the highest performing rules for all tasks with the exception of the dense variant of the rectangle pixel bounding task. Table 5.1 shows higher measured performance of the GA-evolved rules for the two-dimensional density classification, the global synchronization and the spatial density niching tasks than the performance of the human designed rules. The evolved rules for the spare variant of the rectangle pixel bounding task have comparable performance to the human designed rules, while the naïve rule outperforms the GA-evolved rules on the dense variant of the same task. Although the search algorithms found rules with similar behavior to the naïve rule for the dense variant of the rectangle pixel

bounding task, the performance of these rules is slightly worse than the performance of the human designed rule.

Spatially extended versions of GA found the next best set of results (Table 5.1 rows 3-4). The final population of candidate solutions had multiple genomes with high performance, but sometimes different behavior. This observation suggests that the spatially distributed populations maintain higher genetic diversity during evolution. There might be several reasons why the highest performing rules were found by the Standard GA; these include: small sampling size (tournament size) for fitness evaluation and reproduction, undesired evolutionary dynamics during search [13, 132], and inaccurate fitness definition for the evolving test cases for the image processing tasks.

	2DCT (%)	GS (%)	SDN (10^{-3})	RPB-SV (10^{-3})	RPB-DV (10^{-3})
Standard GA	83.29	95.51	60.00	0.77	1.75
NonSpatial CoEv	52.79	79.72	2.10	0.09	0.93
Spatial GA	72.61	88.55	35.14	0.64	0.80
Spatial CoEv	72.17	91.35	22.48	0.59	0.82
2DGKL Rule	58.71	0	4.60	0.75	0.75
Naïve Rule	0	0	3.70	0.75	6.80

Table 5.1: Measured performances of the best GA-evolved rules found for a given task by a given search algorithm. The performance of the GA-evolved rules is listed in rows (1-4), while the last two rows list the performance of the human designed 2DGKL and the the naïve (local majority) rules. The columns (from left to right) list the rules' performance on the two-dimensional density classification (2DCT), global synchronization (GS), spatial density niching (SDN), rectangular pixel bounding – sparse variant (RPB-SV), and rectangular pixel bounding – dense variant (RPB-DV) tasks.

The performance values in Table 5.1 were measured on a randomly initialized 10^4 ICs with the size of 29×29 cells for 600 iterations. The test cases for the classification tasks were generated at random with the binomially distributed IC density of 0.5. The tests for the spatial density niching used randomly generated

foreground and background densities that varied by at least 30%. The sparse variant of the rectangle pixel bounding had test cases generated with density around 10%, while the task's dense variant used test cases with 65% density. The performance values for the classification tasks are given as a percentage value, while the rules' performance on the image processing tasks is in a pixel distance matric – $(pixel\ distance)^{-1} \times 10^{-3}$. (See Section 4.3 for the rule's fitness and performance definitions as well as the details of the experimental setup).

5.2 RULE BEHAVIOR AND PERFORMANCE

Each task section in this chapter contains an analysis of a rule's behavior that is entirely based on the exhibited space-time dynamics. Since all rules solve tasks by global collective behavior, explaining this behavior by analyzing individual bits in the LUT would be difficult, if not impossible. Instead, the discussion of the lattice dynamics should provide an intuitive understanding of the rule's operation. It should not be mistaken for a formal nor final explanation of the information processing in the lattice. It is marely provided as a commentary on what seems to be happening in the lattice. The description of the characteristic behavior summarizes the pattern of behavior that was observed over many CA runs with different random initial configurations.

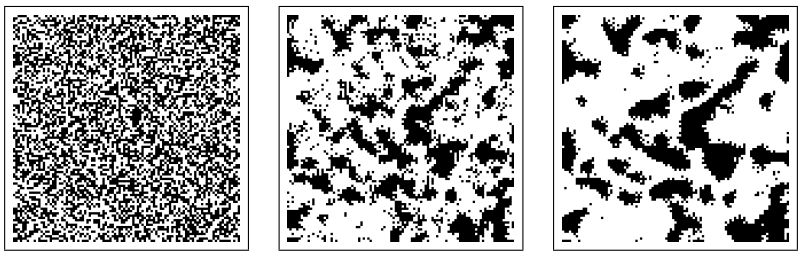
Land and Belew showed that there is no perfect solution to a density classification task using only one rule, with arbitrary neighborhood radius, in any dimension [66]. Similarly, there is no evidence that any of the remaining tasks have a perfect one rule solution. Despite this fact, the genetic algorithm found multiple high quality rules that solve the proposed tasks.

A task solution is represented by the final lattice configurations. Deciding if a solution is correct is trivial for the classification tasks. A solution is correct if the lattice converged to a desired configuration, otherwise a solution is incorrect (no partial credit is given). On the other hand, the image-processing tasks do not

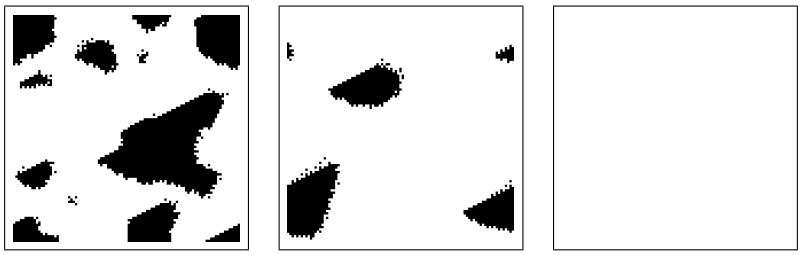
have a binary notion of correctness. Instead, a solution is considered correct if the final configuration "strongly resembles" a desired output.

5.3 DENSITY CLASSIFICATION

For the density classification task, the GA evolved rules with high classification accuracy, and the rule's performance correlates with its behavior. The high performing rules have clearly defined information carrying structures, and the outcome of interactions among the domains is easily predictable. The behavior of rules with low performance has less structure and the interactions are ambiguous.

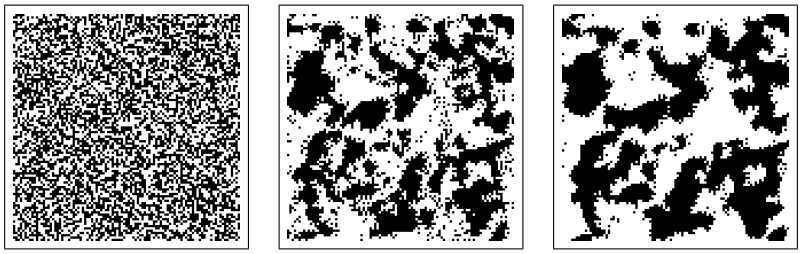


(a) $t = 0$ (b) $t = 5$ (c) $t = 10$

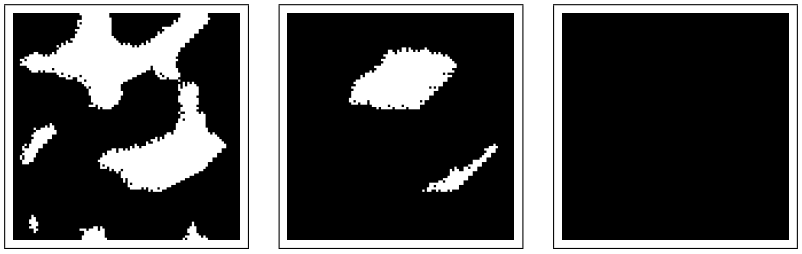


(d) $t = 50$ (e) $t = 150$ (f) $t = 286$

top:



(g) $t = 0$ (h) $t = 5$ (i) $t = 10$



(j) $t = 50$ (k) $t = 100$ (l) $t = 161$

bottom:

Figure 5.1: A series of space-time diagrams of a density classification rule evolved by the genetic algorithm on a 99×99 lattice with $r = 1$. **(a)** The initial configuration has a majority of 0s. **(b) - (f)** Left to right snapshots of CA at evaluation 0, 5, 10, 50, 150, and final configuration at evaluation 286. **(g)** The initial configuration has a majority of 1s. **(h) - (t)** CA at iterations 0, 5, 10, 50, 100, and CA converged to all 1s configuration at evaluation 161.

Figure 5.1 shows the behavior of a 2DCA rule evolved for the two-dimensional density classification task using the Moore neighborhood with $r = 1$ on two initial configurations (top two rows: majority of cells are white; bottom two rows: majority of cells are black). The space-time behavior of the 2DCA shows the creation of black and white domains and their movement. The regions appear to be well defined by their boundaries, and a region's motion can be described as an advancement of its boundary. The observation that a domain's movement can be characterized as an advancement of its bounders may suggest that the domain boundary regions capture the mechanism of information processing in 2DCAs. The results of the filtering methods presented in Chapter 7 highlight the sites that form the domain borders as the sites that are "informationally relevant". These results strengthens the hypothesis that the domain borders are the information-carrying structures in 2DCA.

The best rule was evolved using the standard GA with a performance of 76.6%. The performance was evaluated on 10^4 ICs of 19×19 cells wide that were randomly generated with 0.5 binomial distribution. Based on the observations of the rules' behavior, the rules can be categorized into four performance classes. Both the rule performance categories and the corresponding performance values with the rule classification are similar in one and two-dimensional lattices [27, 51].

The following are the performance class categories:

- **Random rule** (performance is between 0% and 35%): Rule does not converge to the desired configuration, and has unstructured random behavior.
- **Default rule** (performance is between 35% and 50%): Regardless of the initial density, the initial lattice always converges to the same configuration of all white or all black.
- **Domain Expanding rule** (performance is between 50% and 70%): A lattice area of all white or all black, that is larger than the neighborhood radius,

expands until it overtakes the lattice.

- **High Performance rule** (performance is between 70% and $\sim$ 90%): The behavior is similar to the domain expanding rule, but the rules have higher performance. The higher performance value is mainly due to the rule's sophisticated interactions between colliding regions. A detail analysis and additional details are given in the following section.

5.3.1 Comparison with other rules

Table 5.2 compares the performance of the best rules evolved by the GA (columns 4-6) with the performance measured using the human-designed rules (columns 2, 3). The GA-evolved rules outperform both the human-designed 2D GKL and the "naïve" rules. Researches Gacs, Kurdyumov, and Levin (GKL) [38] designed the original 1DCA rule for the density classification task, and the definition of 2D GKL rule is derived for the two-dimensional variant of the density classification task as following: if a cell's state is 0, then change its state to the majority state among its neighborhood's Central, North, and East cells; if cell's state is 1, then change its state to the majority of neighborhood's Central, South, and West cells [55]. I defined the "naïve" rule as a simple majority rule as following: update the cell's state to the majority of a neighborhood's configuration. The performance of all tested rules gradually decreases with the increasing lattice size, since the imperfect rules make more errors on larger lattices. Even though the GA used relatively small lattices (around 20×20) to evolve CA rules, the performance of these rules decreases with increasing lattice size at the comparable rate to the performance decrease in human-designed rules.

Although Cenek's rules evolved by the standard GA has lower performance (Table 5.2 column 4) in comparison to the best rule found by Wolz & de Oliveira (column 5) and Marques-Pita (column 6) [14, 140], they are the simplest. The

CA size	2D GKL	Naïve Rule	Cenek	Wolz & de Oliveira	Marques-Pita
9×9	65.37	51.01	80.21	85.99	87.37
19×19	61.63	49.87	76.60	82.56	85.50
29×29	59.48	50.45	72.71	79.95	83.57
39×39	58.93	50.30	70.87	75.89	82.00
99×99	53.88	50.22	60.05	52.32	76.49

Table 5.2: The performance scaling of human-designed LUT (2D GKL), naïve rule, Cenek's, Wolz & de Oliveira's, Marques-Pita's rules for the 2D density classification task. The performance was measured as a percentage of correctly classified 10^4 random ICs generated according to a binomial distribution. (See Appendix A for binary representations of each of these rules.)

simplicity of a rule is in its symmetry and in its behavior.

The definition of a rule's symmetry is the presence of a pattern of the output bits in the LUT that is repeated through the rest of the table. The longer the pattern of repeated bits, the higher the rule symmetry, and vice versa. Marques-Pita found the rule with the highest classification accuracy. The structure of this rule has low symmetry, in comparison to Wolz & de Oliveira's rule or Cenek's rule. The tradeoff between the gain in the computational performance and the rule's structure should be considered if the CA is implemented as a circuit. Rules with less symmetry will result in devices with more gates and wiring, which might or might not be worth the gain in the computational performance.

Figure 5.2 shows typical behavior of Cenek's [14], Marques-Pita's [14], and Wolz and de Oliveira's CA update rules [14, 140] for the two-dimensional density classification task. The CA rules were applied on the same random initial configuration with the majority state 1 (50.86% initial density). After the initial $5 - 15$ updates, the CA lattice settles into regions of black, white, checkerboard, and striped patterns, also referred to as "domains." The domains, with their simple repetitive patterns, can be thought of as storing information [44, 45, 52, 71, 115]. The domain borders can be thought of as information-carrying "particles," which

move in space and time. (The text will refer to the interface between two domains as a "particle" to be consistent with the 1DCA terminology). The particles become apparent after the domains are subtracted from a CA's spatio-temporal behavior [52].

(a) Cenek [14] (b) Marques-Pita [14] (c) Wolz and de Oliveira [14, 140]

Figure 5.2: Typical lattice configurations produced by rules evolved by genetic algorithms for the two dimensional density classification task on a 99×99-cell lattice at time $t = 20$, for the same random initial configuration. (a) Cenek's rule. (b) Marques-Pita's rule. (c) Wolz and de Oliveira's rule. Highlighted features illustrate the characteristic behavior of the rules. Feature 1 represents a domain boundary that moves in different directions at varied velocities, Feature 2 points to "noisy" borders where the edge of the boundary is not clearly defined. Feature 3 illustrates single-cell-wide domains. Feature 4 highlights the borders between multiple domains that have the same pattern (stripes) but move in different directions.

Each of the rules has very different behavior in terms of the structure of domains, interactions among domains, and border dynamics. In particular, the domain structures are fairly complex; the different sections of a domain's edge can move in various directions at non-uniform speed (Figure 5.2, feature 1); the domain borders are "noisy" and not clearly defined (Figure 5.2, feature 2); a domain can be only one cell wide (Figure 5.2, feature 3); and an edge will be present between two identical and adjacent domains when they move in opposite directions (Figure 5.2, feature 4). Moreover, possible outcomes of collisions among domains include one domain overtaking the other domain (absorption), the creation of a domain with unique structure (generation), or pairs of domains moving through

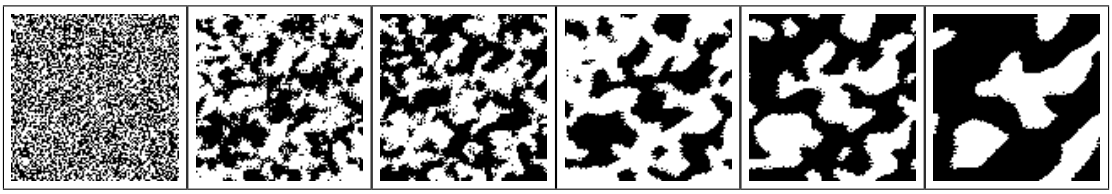
each other (permeation). A domain absorption is best illustrated by a collision between a single-cell-wide ("single-wall") domain of black cells (Feature 3) and a black domain where the single-wall domain is dissolved on impact. A new domain is generated, for example, when a single-wall domain of white cells collides with a striped domain and creates a new domain of all white cells. Finally, a permeation is best visible in a collision between a striped domain and a black domain. The black domain travels through and is surrounded by the striped domain. The domain interactions are not clearly defined (which makes it hard to distinguish the domains involved in a collision from the noise generated by the interaction) when single-wall domains are too small and infrequent (which makes them difficult to differentiate them from noise), and when the domain pattern is not different enough to recognize domain borders (e.g., when two neighboring domains with the same pattern definition share an edge because they move in opposite directions).

5.4 GLOBAL SYNCHRONIZATION

Figure 5.3 shows typical behavior of a rule that solves a global synchronization task on three instances with the initial configurations density around (i.) 50%, (ii.) 30%, and (iii.) 70%. Although the GA-evolved rules for this task have very similar behavior to the rules that solve the two-dimensional density classification task, the performance of many rules exceeds 90%. An intuitive explanation for such high performance is that while the rules for the global synchronization task process information in similar fashion to the rules for the density classification task, unlike the density classification task, the global synchronization task does not have a "wrong" configuration to converge to with respect to the IC's starting density. In other words, as long as the rules converge to the oscillation of black and white configurations, the solution is correct. The only time that the rules failed to solve the global synchronization task was when the lattice consisted of partially black and white regions with zero velocity borders (stagnating regions that failed

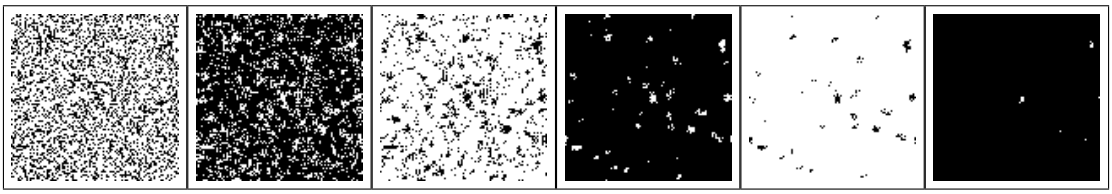
to converge by the maximum time $M = 900$).

Recently published results by Oliveira et al. present global synchronization rules with 100% performance for some lattice sizes. The authors make a conjecture that the non-convergent rules fail to settle into the desired configuration because there exists another cyclic state that attracts the converging lattice and will not allow for further synchronization [94].



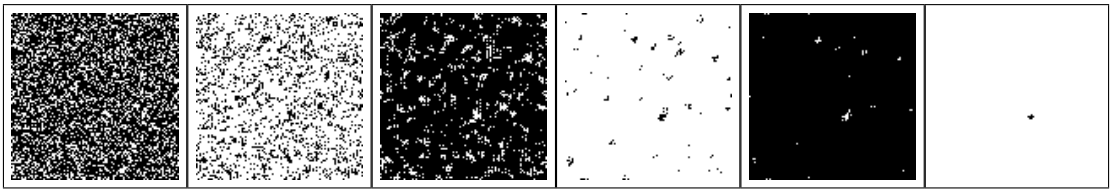
(a) $t = 0$ (b) $t = 5$ (c) $t = 6$ (d) $t = 15$ (e) $t = 20$ (f) $t = 50$

i. Global Synchronization task on IC with initial density 49.83%.



(g) $t = 0$ (h) $t = 1$ (i) $t = 2$ (j) $t = 5$ (k) $t = 6$ (l) $t = 9$

ii. Global Synchronization task on IC with initial density 30.25%.



(m) $t = 0$ (n) $t = 1$ (o) $t = 2$ (p) $t = 5$ (q) $t = 6$ (r) $t = 9$

iii. Global Synchronization task on IC with initial density 69.22%.

Figure 5.3: A series of space-time diagrams of 99×99 lattice for global synchronization task. The initial configuration is shown at time $t = 0$, and the rule converged to the oscillation of all black and all white configurations (not shown for brevity). Behavior of the same GA-evolved rule applied on three different ICs with density **(i.)** 49.83%, **(ii.)** 30.25%, and **(iii.)** 69.22%. The lattices converged to oscillating configurations of all black and white at time-step **(i.)** 228, **(ii.)**, 12, and **(iii.)** 13.

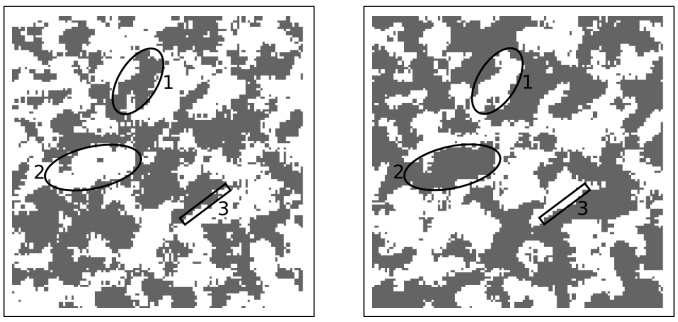
Similarities between the rules evolved for the global synchronization task and

the two-dimensional density classification task are as follows: both lattices settle into small regions of black and white, these regions expand and contract over time, the regions interact with each other at the place of contact, and finally the lattice settles into the final configuration(s). The only difference is that the desired oscillation between all black and white is assumed early on. The lattice settles in to small regions of black and white, and, in the next time-step, these regions reverse their color (black changes to white and vice versa). This behavior is illustrated by the regions marked 1 and 2 shown in Figure 5.4. Despite the regions assuming the opposite state, the regions' boundary continue to advance though the lattice. Feature 3 in Figure 5.4 points out the motion of a domain border by one cell in the south-east direction.

From the observations of lattice behavior, it appears that the explanation of the task's difficulty stated in the previous section also captures the mechanism by which rules solve this task. First, the rules synchronize the local neighborhoods to all black or white, then they propagate these regions through the lattice by enlarging or shrinking their area. The motion of these regions serves as a signalling mechanism to synchronize all, non-adjacent, regions in the lattice to a common phase.

As a side-note and a basis for further discussion in Chapter 10, an apparent symmetry and structure of the LUT has to be further explored. This initiative originates from an unplanned scientific experiment, that resulted from using an existing rule for the two-dimensional density classification task in reverse, and realizing that it also solves the global synchronization task. Closer examination of the LUTs for both tasks revealed that the rules are assembled from repeating motifs of bits. Similar observations about rule symmetry were reported by Oliveira et al. and Marques-Pita [81, 94].

Complementing the n most significant bit(s) and the n least significant bits in the rules for 2D density classification task was not enough to create rules that



(a) $t = 5$ (b) $t = 6$

Figure 5.4: A typical behavior of rules for the global synchronization task shown on an initial configuration with density 49.83% at time $t = 0$. Two consecutive lattice configurations are shown at time-steps **(a)** $t = 5$ and **(b)** $t = 6$. The circles 1 and 2 show transformation of an all black region to an all white region and vice versa. Rectangle 3 highlights a border between white and black regions that moves from one time step to another.

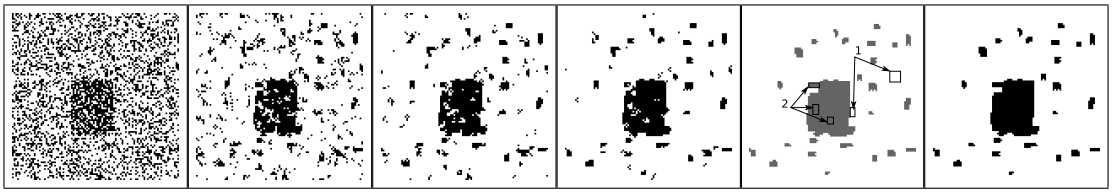
solve the global synchronization task. Instead, each motif (repeated pattern of lookup table bits) had to be reversed to create a rule with a desired effect. This observation leads me to believe that a rule's structure and symmetry could be related to its functionality and performance.

5.5 SPATIAL DENSITY NICHING

Although the GA used CA with a wrap-around boundary condition, the behavior of the evolved rules for the spatial density niching task does not have structures propagating through the lattice over large distances. The rules do not have a global or a lattice-wide collective behavior. Instead, the information-carrying structures propagate information over short distances. Since the "density niches" in this task do not take up the entire lattice, local behavior is sufficient to solve the problems. This being said, the completion of the tasks still required cooperation of cells beyond the neighborhood radius.

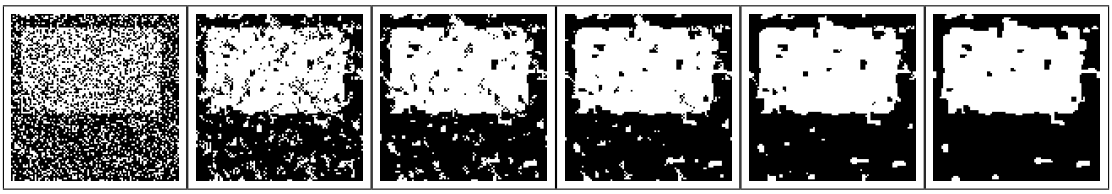
The cells cooperate on a lot smaller scale. After the domains are formed they

expand or contract for around 20 time-steps (also referred to as erosion and deposition). The total number of cells that eroded or were deposited over time exceeds the neighborhood diameter. This signifies the transfer of information beyond a cell's connectivity radius.



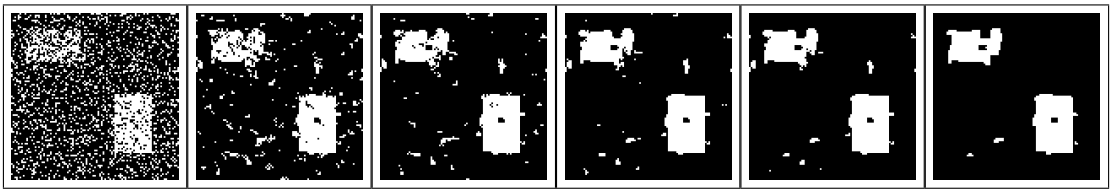
(a) $t = 0$ (b) $t = 3$ (c) $t = 5$ (d) $t = 7$ (e) $t = 10$ (f) $t = 20$

i. Positive task variant.



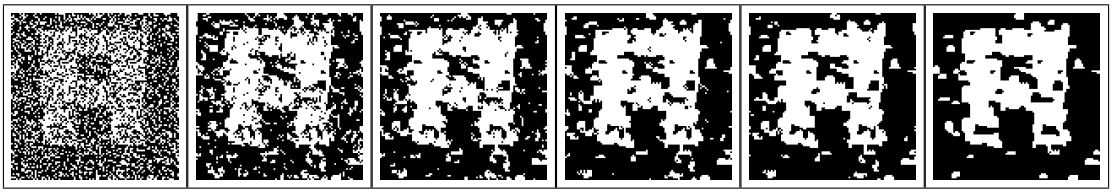
(g) $t = 0$ (h) $t = 3$ (i) $t = 5$ (j) $t = 7$ (k) $t = 10$ (l) $t = 20$

ii. Negative task variant.



(m) $t = 0$ (n) $t = 3$ (o) $t = 5$ (p) $t = 7$ (q) $t = 10$ (r) $t = 20$

iii. Mixed task variant.



(s) $t = 0$ (t) $t = 3$ (u) $t = 5$ (v) $t = 7$ (w) $t = 10$ (x) $t = 20$

iv. Non-trivial polygon.

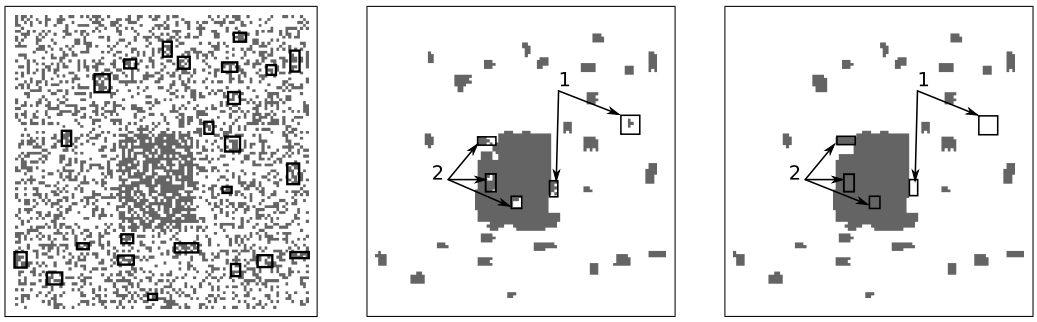
Figure 5.5: A series of space-time diagrams of a spatial density niching rule evolved by the genetic algorithm on a 99×99 lattice with $r = 1$. **(i.)** Initial configuration for a positive task variant with a 67.87% density rectangle on a 38.11% background. **(ii.)** A negative task variant configuration with a 33.09% density rectangle in a foreground and a 68.06% density background. **(iii.)** Mixed niche variant with two rectangles of different densities. The top left quadrant of the lattice has a 19×32 rectangle with density 49.51% and a 35×22 rectangle with density 19.92% in the bottom right. **(iv.)** A solution for a non-trivial polygon. The 'A' shape foreground has a density 42.87% placed on the background with 65.46% density.

Figure 5.5 shows how the same rule solved all three variants of this task. For each variant, during the first three steps, the CA lattice settles into small regions of black and white. The rest of the CA iterations shrink or enlarge the black and white domains based on their location. In each case the final configuration is reached around $t = 20$. Although not a fixed point, the convergence times are similar and only vary when a lattice size changes. Figure 5.5 (i.) shows results on the task's positive variant. The domains formed by rules evolved for the density classification task shrink or grow by advancing a clearly defined domain border in space and time. The domains in the density niching task alter their shape and size in a chaotic, disorganized, or noisy fashion. The edge of the domain slowly erodes by a seemingly random subtraction of black pixels. The growth of a domain is equally noisy, and looks like a deposition of black pixels on a surface.

The reason that the final lattice configurations look like Swiss cheese is because the process of shrinking or expanding is done on the small domains that formed early on. The erosion and deposition of pixels happens for a short period of time (usually 5-15 steps), which means that not all undesired rectangles disappear from the final configuration. The process of shrinking and growing domains can be seen in Figure 5.6 (b.) and (c.) Ten lattice updates separate these two images, where feature 1 represents slow erosion and feature 2 points to pixel deposition.

It is worth noticing that the shapes of the remaining structures in the final configuration also resemble rectangles. Figure 5.6 a. shows the outline of the rectangles from the final configuration at the initial configuration. The average starting density inside of these rectangles is 55.56%, considerably higher than the background density of 38.11%. This might explain why the CA also highlighted these secondary niches in the final configuration.

Although GA found rules that work on all positive, negative, and mixed definitions of this task (Figure 5.5 i., ii., iii. respectively), they fail to find the density niche if the difference between the background and foreground is less than 20%.



(a) $t = 0$ (b) $t = 10$ (c) $t = 20$

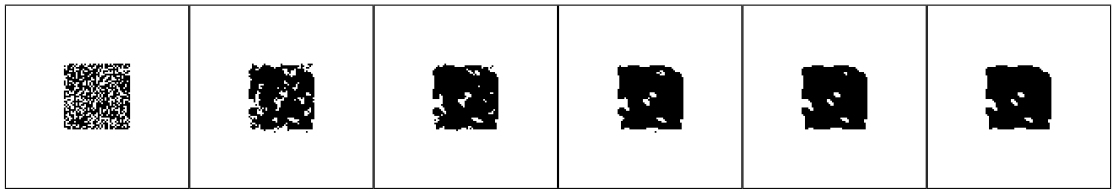
Figure 5.6: Typical behavior of rule on positive density niching task. The higher density rectangle of 67.87% is embedded in a 38.11% background. **a.** The overlay of black rectangles on the initial configuration represents small domains that remain in the final lattice configuration (shown in **c.**). The average starting density inside the outlined rectangles is 55.56%. The images in **b.** and **c.** show slow, noisy shrinking and growing of domains. Feature **1** represents erosion while Feature **2** points to domains that grew.

The GA evolved rules used IC training configurations with a minimum of 30% density difference; the search failed to evolve rules for any of the task variants if the density difference in the training ICs was less than 30%. Finally, Figure 5.5 iv. shows the results on an initial configuration with a non-trivial polygon. The CA outlined the desired shape, despite the fact that the rules were evolved using only training ICs with simple rectangles.

The mixed task variant can be solved only if the background is the lightest or the darkest out of the three densities in the IC. In other words, the CA fails to detect both the light and dark rectangle if the background density is somewhere in the middle. An example of such an IC would have a background with 50% density, and one rectangle with 30% density and a second rectangle with 70% density. Figure 5.5 iii. shows the result of the rule applied on an IC with two rectangles with densities of 19.92% and 49.51% on a background with 79.98% density.

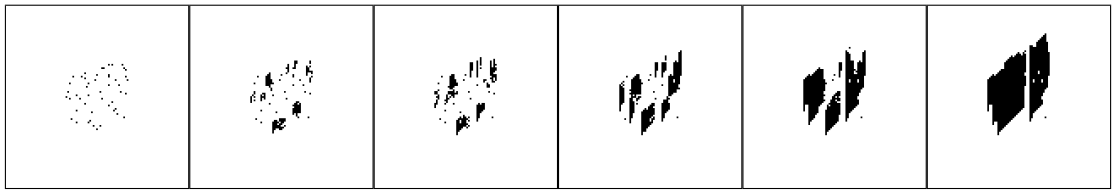
5.6 RECTANGLE IMAGE BOUNDING

When evolving rules for the rectangle image bounding task, the GA found many rules with a similar behavior to the rules for the density classification task. These rules achieved good quality results on the dense image bounding task, but images with very sparse initial configurations were turned all white. The typical behavior of these rules can be described as a default domain rule — a region with majority black converges to all black, while regions with few pixels turn all white. Figure 5.7 i. shows the behavior of such a rule. The rules that solve the sparse variant of this task had to be evolved separately using images with sparse pixels as training cases. Typical behavior of rules for sparse image bounding can be seen in Figure 5.7 ii.. Figure 5.7 iii. shows the behavior of a rule evolved for sparse image bounding applied to a dense IC.



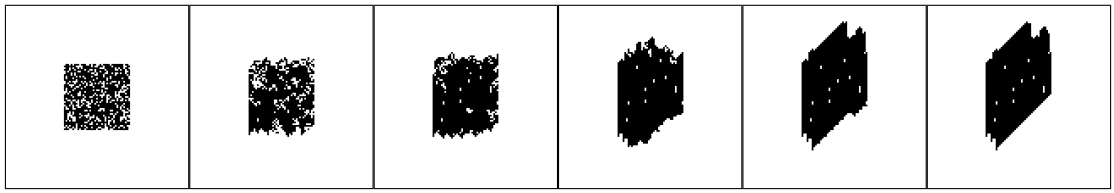
(a) $t = 0$ (b) $t = 3$ (c) $t = 5$ (d) $t = 7$ (e) $t = 10$ (f) $t = 20$

i. Dense image bounding - Default domain rule.



(g) $t = 0$ (h) $t = 10$ (i) $t = 20$ (j) $t = 50$ (k) $t = 100$ (l) $t = 200$

ii. Sparse image bounding.



(m) $t = 0$ (n) $t = 10$ (o) $t = 20$ (p) $t = 50$ (q) $t = 100$ (r) $t = 200$

iii. Dense image bounding.

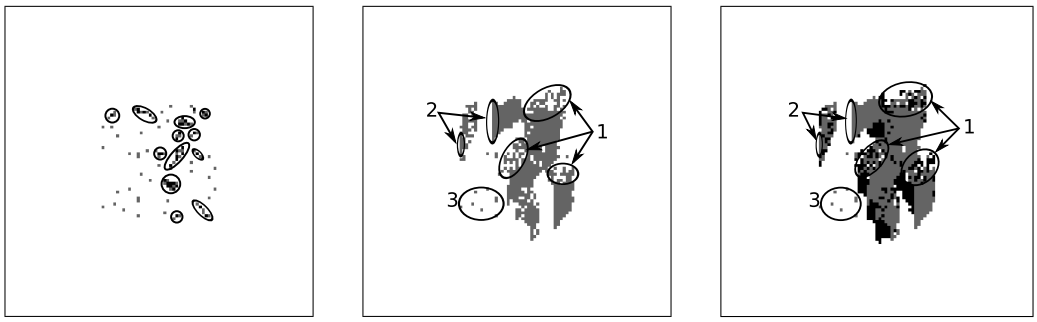
Figure 5.7: A series of space-time diagrams of a 99×99 lattice for the sparse and dense image bounding task. The initial configuration is shown at time $t = 0$, and the final configuration is captured as the final image in each of the series. **i.** A commonly found rule for the dense image bounding task with a default domain behavior. The initial configuration has a rectangle with 61.46% pixel density. This rule failed on the sparse image bounding task. **ii.** A sparse image bounding task variant with 3.20% pixel density. **iii.** A dense image bounding task variant with 72.26% pixel density. The same rule was used for sparse and dense bounding task configurations in **ii.** and **iii.**.

The rules that were able to solve both problem variants have the most complicated dynamics out of all evolved rules for any of the tasks defined in this chapter (Figure 5.7 ii. and iii.). Even though the final lattice configuration is not a perfect polygon, a coherent black domain is formed containing the IC pixels. The dense image bounding task variant is best solved by the default domain rule, since the sparse image bounding rule misclassifies large number of the IC pixels (Figure 5.7(i.) and (ii.) respectively.). The term "to solve a task" has a less strict meaning for the rectangular image bounding solutions than for the rest of the tasks.

Although the overall behavior of these rules can be described as domain expanding, it is not clear what the exact mechanism of domain expansion is. The domain expanding rules that solve the density classification task show structured domains that travel though space and interact with other domains. The rules for the spatial density niching task, although noisy, clearly form black and white regions and the mechanism of domain shrinking and growing is also visible. The behavior of the rules for the sparse rectangular image bounding task has the least structured domains, and the mechanism of expansion has no obvious pattern.

In comparison to the GA-evolved rules, the local majority rule will also converge a lattice to a configuration that is similar to the ideal output for the dense image bounding task. Its overall behavior is very similar to the default domain rule (Figure 5.7 (i).). The rule fails to converge on the sparse image bounding task and it turns all black pixels in the IC white.

The behavior of the rules that solve sparse image bounding is as follows: If a local neighborhood configuration contains at least two black pixels, they act as a seed for the domain expansion (Figure 5.8 a). The domain starts expanding in a North-South or East-West direction. The length of expansion is short, typically only 5-10 time-steps. If the growing domain intersects another domain, the two domains merge and keep expanding as one cluster. If a domain fails to merge with another domain, the domain stops growing and becomes constant (motionless).



(a) $t = 0$ (gray), $t = 1$ (black)

(b) $t = 40$

(c) $t = 40$ (gray), $t = 50$ (black)

Figure 5.8: A typical behavior of rules for the sparse rectangular bounding task. The bounding box of the black pixels has 4.87% density. **a.** The circles in the initial configuration show the seed locations for expanding domains. Gray pixels mark the initial configuration, while black represents newly generated pixels at time $t = 1$. **b.** Time step $t = 40$ with Feature 1 highlighting examples of active fronts. Feature 2 points to the constant domain walls. The pixels from the initial configuration that were not reached are marked as feature 3. **c.**) The lattice configurations at time $t = 40$ is represented by gray pixels while black pixels mark time $t = 50$.

The only way for a domain to start growing again is if it merges with another actively expanding or contracting domain. The merger 'fuels' the domain's growth. The domain expansion period is short if the growing front is only a few cells wide. A domain with a wide expansion front stays active for longer periods of time than the front in narrower domains (Figure 5.8 feature 1).

The velocity of domain expansion is extremely slow. The speed of the domain borders in rules evolved for density classification was between 0.5 and 2 sites per update (estimated from observations). The rules evolved for the sparse image bounding task variant have domains that expand at half the rate. This phenomenon is possible because the domains expand by a noisy front (Figure 5.8 features 1). This front with non-trivial structure advances with low velocity, and unless it recombines with another domain, the front will slowly turn to all black and stops advancing. The slow domain expansion is shown as a difference between time step $t = 40$ and $t = 50$ in Figure 5.8 c.

Since domains expand in orthogonal directions, the sides of the domain perpendicular to the direction of growth do not expand. The examples of constant domain walls are highlighted as feature 2 in Figure 5.8. This growth behavior limits the width of a domain. If the horizontal spacing between the domains growing in the North-South direction is less than the neighborhood radius, the domains will never merge and cause a gap in the final configuration (Figure 5.7 ii. (l) and 5.8 c).

5.7 SUMMARY

All the proposed tasks challenge the ability of CA to exhibit cooperative behavior among the lattice cells. Although the rules for the two-dimensional density classification, the global synchronization and the rectangle image bounding have different characteristic behavior, they all have structures with size exceeding the neighborhood radius. The propagation of these structures through the lattice acts as a signalling mechanism that communicates the information about local configuration beyond a cell's connectivity radius. Since a solution to a given task is determined by the convergence of the entire lattice to a desired state, the interactions among multiple information signals function as a mechanism of lattice convergence. Since the rules for the image processing tasks rarely (if ever) converge to the ideal output configuration, the notion of a "solution" was relaxed to convey a strong similarity between a final lattice configuration and the ideal output. The statement that the GA-evolved rules "solve" the image processing tasks is more accurate using this less strict definition of success.

The GA-evolved rules for the spatial density niching task are the only rules that do not show a behavior where the global cooperation among cells creates a lattice-wide patterns. Over time, the process of domain erosion or deposition causes the lattice to converge into a final configuration. Although the rules' behavior does not have lattice-wide patterns, the lattice communicates the information about a local

neighborhood beyond the cell's neighborhood radius. Even though no lattice-wide patterns are formed, the cells have to cooperate past their local connectivity to ensure convergence.

The focus of this research is not to find the best performing rules to solve these tasks, but to test if the CA is capable of performing collective computation on tasks other than the two-dimensional density classification task. The GA evolved high performance rules for the two-dimensional density classification task as well as rules that solve the other proposed tasks. Although no perfect solutions were found for the image processing problems, the chapter's previous sections presented the GA-evolved rules that converge to final lattice configurations that strongly resemble the tasks' ideal output.

The behavior of the evaluated rules ranges from simple, such as rules for the density classification task, to rules with complicated behavior where even a human observer has a hard time pinpointing the mechanism of lattice convergence. The solutions for the rectangular image bounding task represent such rules with non-trivial signalling behavior. The information carrying domains in rules for the density classification task travel through the lattice until they interact with another domain wall or until they cease to exist. The length of the information carrying signals is much shorter in the rules evolved for the rectangular image bounding task. This is mainly because the feature size does not take up the entire lattice, so the information processing signals in the rules have much shorter activity period (lifespan) and move for a limited distance (range).

In this chapter the behavior of the evolved rules is explained in colloquial and intuitive terms. Such imprecise language is used because the CA behavior can be described only informally, since no link has been established that connects the occurrence of the lattice-wide patterns and the CA's computational mechanics. Even if a model of computation in 2DCA would confirm such connection, any such model that abstracts and generalizes the mechanism of collective computation

in the lattice from the space-time diagrams should take into consideration the empirical nature of underlying analysis. This is the main reason for the vague description of a rule's behavior, and is a restraint to explaining CA behavior as the mechanism of computation in 2DCA.

Chapter 6

INFORMATION PROCESSING VIA PARTICLES[1]

Although Chapters 1, 2, and 3 intuitively defined terms such as information, processing, CA behavior, and information carrying structures, this chapter provides a more formal discussion of information processing in CA. In addition to providing background information, the following sections set up a contextual framework for Chapters 7 and 8.

Since the topic of collective computation is one of the main topics of this thesis, the first section provides more details on this topic. The second half of this chapter examines the concept of domains and particles as the mechanism of computation in 1DCA. One way of verifying the conjecture that the statistically based filters correctly identified the information carrying structures is to build a model that detects particles and uses them to predict the lattice execution without using the rule-table updates. A detailed explanation of such model for 1DCA is provided at the end of this chapter. Although the construction of a 2DCA model of information processing is analogous to the 1D case, the details of the 2D model are deferred until Chapter 8.

6.1 COLLECTIVE COMPUTATION IN CELLULAR AUTOMATA

A CA capable of universal computation, such as the game of Life and elementary CA 110, can in principle carry out any computation. To actually do a computation using such a CA, one needs to provide in the initial configuration bits that encode

[1]PORTIONS OF THIS CHAPTER WERE ADAPTED FROM [76]

both a "program" and the program's input. Such CAs compute by simulating a Turing machine or similar computing model, running the program on the input by, for example, creating logic gates out of CA configurations such as *blinkers*, *gliders*, and *glider guns*.

Such a simulation is typically a slow process in which a massively parallel machine is used to simulate a serial machine in a highly inefficient and impractical way. While universal computation in simple CAs is a theoretically interesting result, this is not a particularly useful notion of computation if one's goals are to design computation in complex decentralized spatially extended architectures or to understand how natural systems compute.

Given such goals, an alternative approach to computation in CAs is to use the complex dynamics and pattern formation behavior of CAs to perform collective, genuinely parallel computations. This approach has been exemplified in work on applying evolutionary computing methods to design CAs to perform computations [5, 6, 17, 88, 126]. In this work, the CA rule plays the role of "program", the initial configuration plays the role of "input", and a later configuration or configurations play the role of "output".

Designing an algorithm to solve problems (such as the tasks proposed in Chapter 3) is comparatively trivial for a system with a central controller, or central storage of some kind, such as a standard computer with a counter register or a neural network in which all input units are connected to one or more hidden units. However, the tasks are nontrivial for a small-radius ($r \ll N$) CA, since a small-radius CA relies only on local interactions.

Let's look at the density classification task as one such task in more detail. It was proved that no finite-radius, two-state CA with periodic boundary conditions can perform this task perfectly across all lattice sizes [66]. Even to perform this task well for a fixed lattice size requires more powerful computation than can be performed by a single cell or any linear combination of cells. Since the 1s can be

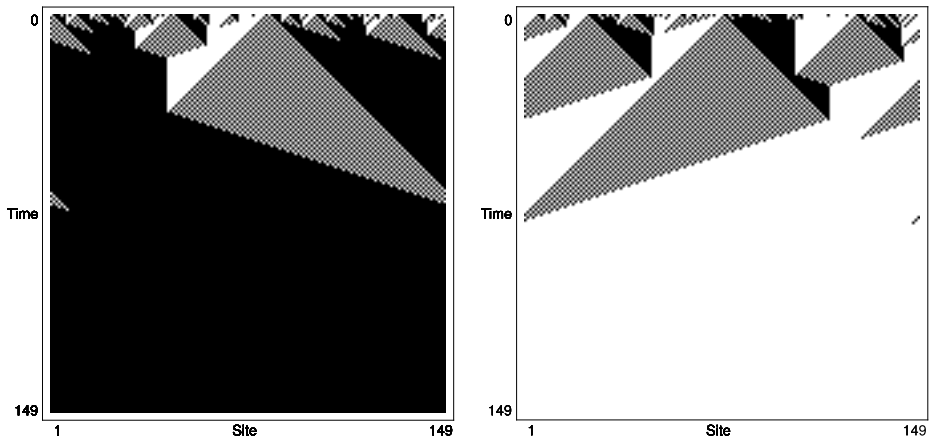


Figure 6.1: Space-time behavior of a CA evolved by the GA for the density classification task [28]. The left diagram shows the CA iterating from a high-density initial configuration (i.e., with a majority of cells in state 1 (black)) and the right diagram shows the CA iterating from a low-density initial configuration (i.e., with a majority of cells in state 0 (white)). In each case the CAs give a correct classification of the initial configuration. This CA correctly classifies about 80% of random initial configurations on 149-cell lattices.

distributed throughout the CA lattice, the CA must transfer information over large distances. To do this requires the global coordination of cells that are separated by large distances and that cannot communicate directly.

Several groups have used genetic algorithms and related evolutionary methods to automatically search the enormous space of binary, radius-3 CA rules to find rules with high degree of accuracy for problems such as density classification (e.g., [2, 57, 88, 140], also see Chapter 3 for more details). A major challenge is to understand the CAs resulting from these searches—that is, to characterize exactly *how* the resulting collective computation is being done by the CAs.

Figure 6.1 illustrates the typical behavior of a CA that was evolved by the GA for the one-dimensional density classification task. The two *space-time* diagrams each plot the one-dimensional lattice configuration versus time for a high-density initial configuration (left) and for a low-density initial configuration (right). Both configurations are correctly classified by the CA's behavior.

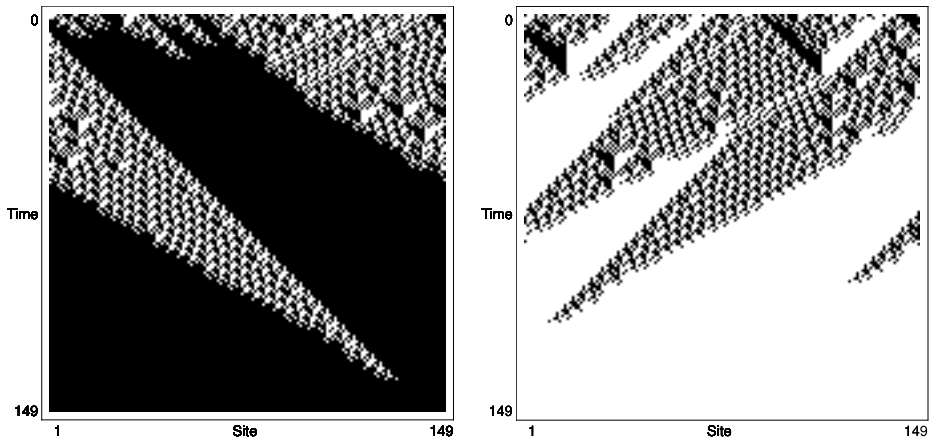


Figure 6.2: Space-time behavior of the highest-performing known CA evolved by the GA for the density classification task [140]. The left diagram shows the CA iterating from a high-density initial configuration and the right diagram shows the CA iterating from a low-density initial configuration. In each case the CAs give a correct classification of the initial configuration. This CA correctly classifies about 89% of random initial configurations on 149-cell lattices.

The CA illustrated in Figure 6.1 has a *performance* close to 80% on one-dimensional lattices of size 149—that is, when given randomly generated initial configurations of the 149 cells, the CA arrives at a correct classification approximately 80% of the time. The highest-performing known one-dimensional CA for this task, also evolved by a GA (see Figure 6.2) has a performance of approximately 89% on 149-cell lattices [140].

Figures 6.1 and 6.2 raise some interesting questions. What are the "algorithms" by which these CAs are achieving their relatively high performance? What gives rise to the higher performance of the CA shown in Figure 6.2? Simply examining the CA look-up tables or even the collection of bits in the space-time diagrams above does not easily shed light on these questions. In fact, traditional computer science, with its assumptions of von Neumann-style architectures, does not provide the necessary tools for understanding the mechanisms of information processing in spatially extended dynamical systems created by evolution, be it natural or, as in our case, artificial. We are left with Stephen Wolfram's final question in his

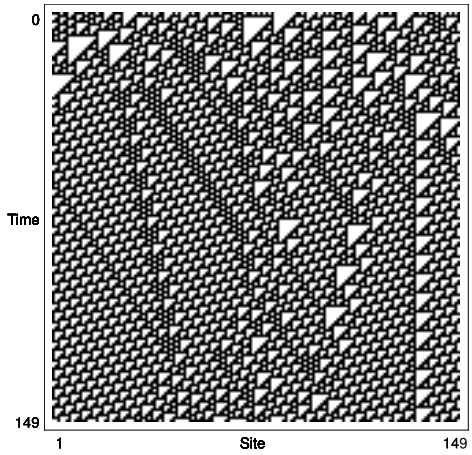


Figure 6.3: Behavior of elementary CA 110 starting from a random initial configuration.

"Twenty Problems in the Theory of Cellular Automata" [137], "What higher-level descriptions of information processing in cellular automata can be given?"

6.2 INFORMATION PROCESSING STRUCTURES: DOMAINS AND PARTICLES

A number of research groups have used the notion of propagating structures called *particles* to characterize the dynamics, computational ability, or mathematical properties of CAs (e.g., [1, 8, 9, 21, 30, 41, 78, 84, 101, 139]), to model natural particle-forming systems (e.g., [16, 99]), as well as to "program" such systems (e.g., [24, 119, 123]). In analogy with the notion of particles in physics, in cellular automata *particles* are localized patterns that retain their coherence while propagating in space and time. A simple two-dimensional example is the so-called *glider* in the Game of Life [8]. In one dimension, examples of particles can be seen in the behavior of elementary CA rule 110 (Figure 6.3); here particles are relatively linear patterns that travel in space and time at various slopes (velocities) against a relatively simple periodic background pattern—a *regular domain*.

Crutchfield and Young [25] introduced the *computational mechanics* framework

for understanding computation in complex systems. Computational mechanics is a set of methods for identifying the "intrinsic computation" in physical systems—that is, how the underlying mechanisms of a physical system support memory and information dynamics, which are the building blocks of "useful computation" that is engineered by humans or by evolution to perform useful functions.

Hanson and Crutchfield applied the computational mechanics framework to the problem of characterizing intrinsic computation in one-dimensional cellular automata. They informally characterized a *regular domain* as "a spatially and temporally homogeneous pattern describable by a finite automaton—where 'homogeneous' is understood in the sense of having the same regularities" [45]. More formally, they define regular domains in one-dimensional cellular-automata space-time patterns as two-dimensional regions consisting of one-dimensional "words" in simple regular languages—those languages that can be represented by strongly connected finite-state automata.

The space-time diagrams given in Figure 6.1 provide simple examples of this notion. The two space-time diagrams show the three regular domains that can be produced by this CA: all black (i.e., all 1s), all white (i.e., all 0s), and a checkerboard-like domain of alternating black and white states. The three corresponding regular languages are $(1)^*$, $(0)^*$, and $(01)^*$ or $(10)^*$. The particles of this system are the spatially localized boundaries separating each pair of adjacent regular domains.

Das et al. [28] pointed out that the density-classification task is equivalent to the recognition of a non-regular language (the equivalent of a counter register is required to track the excess of 1s, so the minimum amount of memory required is proportional to $log(N)$). Thus the computation required to perform the task is not being done in the simple regular domains; it is being done via the boundaries between those domains—that is, by the particles.

Intuitively, here is how the CA illustrated in Figure 6.1 performs the density

classification task. Over short times, local high-density regions are mapped to all 1s, local low-density regions are mapped to all 0s, with a vertical boundary in between them. This is what happens when a region of 1s on the left meets a region of 0s on the right. However, when a region of 0s on the left meets a region of 1s on the right, rather than a vertical boundary being formed, a checkerboard region (alternating 1s and 0s) is propagated. When the propagating checkerboard region collides with the black-white boundary, the inner region (e.g., each of the white regions in the left-hand diagram of figure 6.1) is cut off and the outer region is allowed to propagate. In this way, the CA uses local interactions to determine the relative sizes of adjacent large low and high density regions. For example, in the left-hand space-time diagram, the large inner white region is smaller than the large outer black region—thus the propagating checkerboard pattern reaches the black-white boundary on the white side before it reaches it on the black side; the former is cut off, and the latter is allowed to propagate and eventually takes over the lattice.

As was discussed in Chapter 2, the black-white boundary and the checkerboard region can be thought of as "signals" indicating "ambiguous" regions. The creation and interactions of these signals can be interpreted as the locus of the computation being performed by the CA—they form its emergent "algorithm" [24].

Figure 6.4 gives the left-hand space-time diagram of Figure 6.1 with the regular domains filtered out, leaving only the particles. Table 6.1 lists the types of regular domains, particles, and particle interactions that appear in all space-time behavior of this CA rule. Hordijk et al. [52] describe how to predict the collective behavior of CAs using their particle catalogs to build a *dynamic model* based on equations, and show that the model's predicted performances (i.e., classification accuracies on the density classification task) and the observed performances are very close. On that basis, Hordijk et al. conclude that the particle-level descriptions of CAs provided by particle catalogs characterize the intrinsic computational capability of

Regular Domains		
$\Lambda^0 = 0^*$	$\Lambda^1 = 1^*$	$\Lambda^2 = (01)^*$

Particles (Velocities)	
$\alpha \sim \Lambda^0\Lambda^1$ (0)	$\beta \sim \Lambda^1 01 \Lambda^0$ (0)
$\gamma \sim \Lambda^0\Lambda^2$ (-1)	$\delta \sim \Lambda^2\Lambda^0$ (-3)
$\eta \sim \Lambda^1\Lambda^2$ (3)	$\mu \sim \Lambda^2\Lambda^1$ (1)

Interactions	
decay	$\alpha \rightarrow \gamma + \mu$
react	$\beta + \gamma \rightarrow \eta$, $\mu + \beta \rightarrow \delta$, $\eta + \delta \rightarrow \beta$
annihilate	$\eta + \mu \rightarrow \emptyset_1$, $\gamma + \delta \rightarrow \emptyset_0$

Table 6.1: Catalog of regular domains, particles (domain boundaries), particle velocities (in parentheses), and particle interactions seen in rule's space-time behavior. The notation $p \sim \Lambda^x\Lambda^y$ means that p is the particle forming the boundary between regular domains Λ^x and Λ^y. (Adapted from [88].)

the modeled CAs.

It should be pointed out that particles in CA behavior, in the sense we have described above, are not always obvious by visual inspection. One example is elementary CA rule 18, illustrated in Figure 6.5 (left). This CA has a single regular domain, $(0\Sigma)^*$, where Σ represents either 0 or 1 [44]. In other words, in this regular domain, every other site is a zero; the remaining sites can be either 0 or 1. Figure 6.5 (right) gives the same diagram with this regular domain filtered out, revealing the "embedded" particles, which have been shown by Hanson and Crutchfield to perform a random walk in space-time [44]. Thus Hanson and Crutchfield's analysis of ECA 18 shows that the CA's "intrinsic computation" is a random walk implemented by embedded particles.

In the examples above we have seen several types of CA particles—those explicitly designed (e.g., Life gliders), those in elementary CAs not designed for anything in particular (e.g., ECAs 110 and 18), and those in CAs evolved by the GA to perform density classification. We have sketched how regular domains and particles

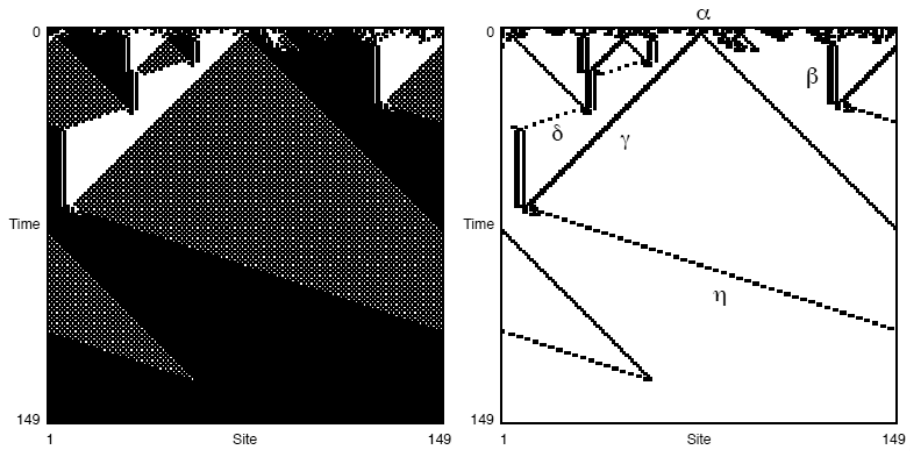


Figure 6.4: (Left) The left-hand spacetime diagram of figure 6.1. (Right) The same diagram with the regular domains filtered out, leaving only the particles (some of which are labeled by here by the Greek letter code of table 6.1). Note that particle α (unlike other the other particles) lasts for only one time step, after which it decays to particles γ and μ.

can provide a highly compressed and useful account of the CA's behavior in computational terms. In general, we would like to find methods that automatically discover such information-processing structures in spatially extended systems, either by appropriately filtering space-time data from these systems or by directly analyzing the governing equations or look-up-table descriptions of such systems. In the rest of this chapter we survey four filtering methods proposed by different groups for achieving this automatic discovery, and we assess how far these efforts take us in our goal to understand and design computation in spatially extended dynamical systems such as cellular automata.

6.3 MODEL OF INFORMATION PROCESSING IN 1DCA

Hordijk et al. were the first to successfully analyze the mechanism of computation in the rules for density classification and synchronization tasks [50, 51, 53].

Figure 6.6 (left) shows one of the CAs evolved for the density classification task

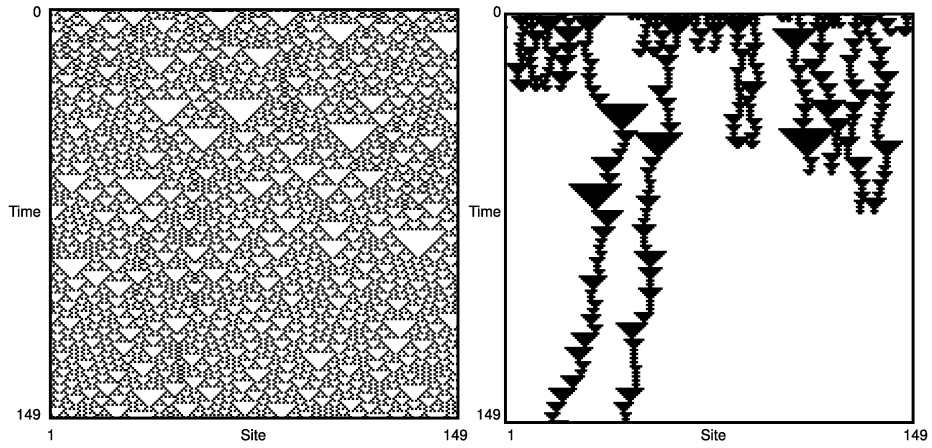


Figure 6.5: (Left) Space-time behavior of elementary CA 18, iterated from a random initial configuration. (Right) The same diagram with the regular domain filtered out, leaving only the particles. (Reprinted from [87].)

similar to the ones analyzed by Hordijk et al.'s dynamic model. The rule forms black, white, and checkerboard domains. The structure of the regular domains can be detected by Hanson and Crutchfield's epsilon machine reconstruction algorithm [23, 25, 43, 45]. After the domains are identified and filtered out, the particles are simplified by straight lines. The intersections of the lines represent the particle collisions. A lookup in the particle interaction catalogue (shown in Table 6.1) describes which particles are substituted as a result in a place of collision (see Chapter 7 for more details). Since, in a model, a particle is represented by a line, a simple vector physics framework is used to simulate particle motion and predict the site and time when an interaction between particles will occur. Hordijk et al.'s model predicted CA behavior with an accuracy of 95% and greater [50, 52].

Recent CA that perform the one-dimensional density classification task are shown in Figure 6.6 (center and right). The rules found by Marques-Pita [79, 83] and Wolz & de Oliveira [140] have higher performance than the rules found by Das et al. [28]. The high-performing rules have more complicated behavior than the lower-performing ones. It is unclear if something like Hordijk et al.'s simplified

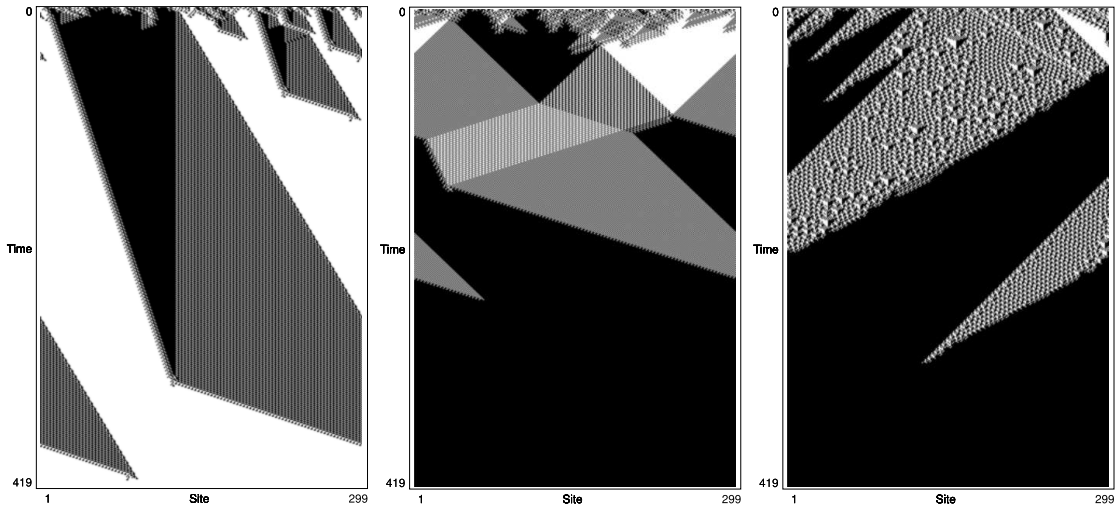


Figure 6.6: GA evolved rules for one-dimensional density classification task for CA with neighborhood radius $r = 3$. The rules were found by Das et al. (left)[28], Marques-Pita (center)[79, 83] and Wolz & de Oliveira (right)[140].

dynamic models is adequate enough to explain the mechanism of information processing in these rules. The model might fail due to the following reasons: A regular language based filtering approach might not be powerful enough to recognize the non-trivial domain structure such as the domain pattern in Figure 6.6 (right). The complicated domain structure might have an additional role of processing information, rather than simply storing information. Simplifying and abstracting the domain borders as line shaped particles might not be correct either. Finally, a vector physics model might have to be reformulated to account for more complicated particle interactions.

To summarize, the dynamic model of Hordijk et al. correctly predicted CA behavior because the CA formed well defined domains, the particles had (roughly) the shape of a straight lines, the particle collisions were easily predicted, and the interactions among particles had clear outcomes. The description of domains, particles, and particle interactions were similar when a rule was executed on different randomly initialized starting configurations. Since the model was built for a specific task and a group of rules with the same behavior, it is unknown how the predictive

power of a model would change for rules with different behavior and rules evolved for tasks other than the density classification and global synchronization.

Chapter 7

FILTERS FOR IDENTIFYING INFORMATION-PROCESSING STRUCTURES IN CA[1]

The first step towards understanding the mechanism of collective computation in a two-dimensional cellular automaton is to identify the sites in the lattice that store, modify, and transfer information. Due to the complex behavior of these systems, building filters to detect such sites by hand is impractical in general. This chapter describes several approaches for automatically identifying the structures underlying information processing in the spatio-temporal patterns formed by cellular automata. In particular, I review the *computational mechanics* methods of Crutchfield et al. [25, 44], the *local sensitivity* and *local statistical complexity* filters proposed by Shalizi et al. [115], and the information-theoretic filters proposed by Lizier et al. [71].

The methods described in this chapter were originally designed for automatic identification of information-processing structures in one-dimensional cellular automata. On a fundamental level, the mechanisms of computation in one-dimensional cellular automata are the same as in two dimensions. The CA behavior settles into domains with well-defined structure, these regions propagate through space, and the interactions among these structures represent processing of information. This chapter first analyzes the behavior and defines the filtering methods for 1DCA, since it is easier to visualize and explain the details in one dimension. My two-dimensional extension of the filters and the filter results on the two-dimensional

[1]PORTIONS OF THIS CHAPTER WERE ADAPTED FROM [76]

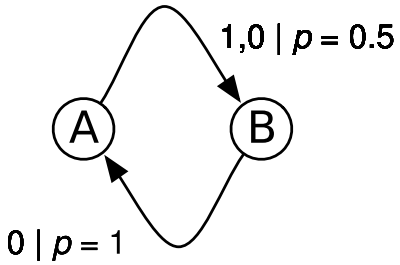


Figure 7.1: Two-state epsilon-machine encoding the "every other site is a zero" regular domain of elementary CA 18.

density classification task are covered in the chapter's second half.

The filtering methods are compared in terms of their computational requirements and their ability to detect spatio-temporal structures in 2D lattices. Finally, a hybrid filtering approach is introduced as a combination of Shalizi et al.'s algorithmic approach with Lizier et al.'s spatial structuring of information flow. The results show that this combination produced the most accurate filtering results while being computationally feasible.

7.1 FILTERING BY EPSILON-MACHINE RECONSTRUCTION

In order to automatically discover regular domains (and thus particles), Crutchfield and Hanson [23, 44] apply the *epsilon-machine reconstruction* algorithm (originally developed by Crutchfield and Young [25]). The result is an *epsilon machine*—a finite-state machine that recognizes regular domains. Epsilon machines differ from deterministic finite state machines in that their transitions are labeled with conditional probabilities (they are essentially equivalent to hidden Markov models [34].)

Figure 7.1 gives an epsilon machine that recognizes the regular domain of elementary CA 18, "every other site is a zero" (cf. Figure 6.5).

An extended version of this epsilon machine—a *transducer* that both inputs and outputs symbols—can be used to filter space-time diagrams containing this

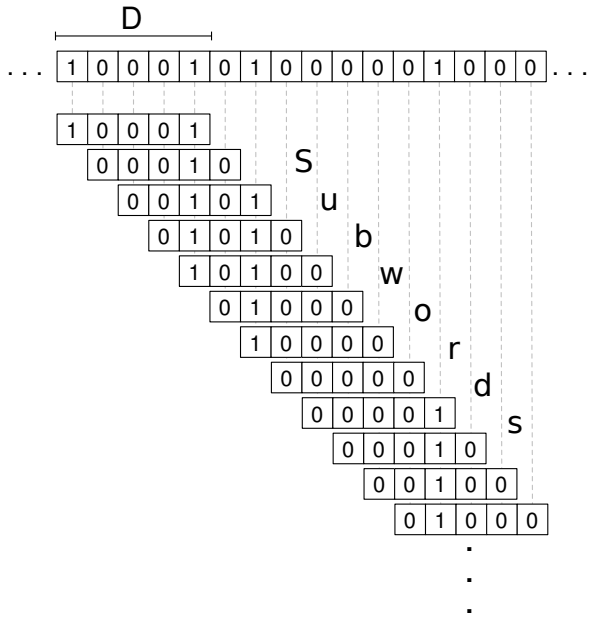


Figure 7.2: Creation of subwords from CA configurations. Adapted from [42].

domain. Such a transducer was used to create Figure 6.5b; details are described in [42].

Hanson [42] gives the following steps for reconstructing an epsilon machine from space-time data generated by a one-dimensional CA.

1. **Generate data:** Run the CA several times starting from a number of random initial configurations. On each run, let the CA iterate for a large number of time steps t to allow short-lived transient behavior to die out. Then collect a set of input configurations from $t + 1$ to some later time step T. Create a set of "subwords" of length D (a parameter of the algorithm) from these configurations. Each subword consists of the symbols inside a sliding window of size D (see Figure 7.2).

2. **Construct tree:** Once a set of subwords is created, they are used to build a tree of depth D, as is illustrated in Figure 7.3. Start with the root node. For each subword, iterate left-to-right through each symbol s. If s is a 0,

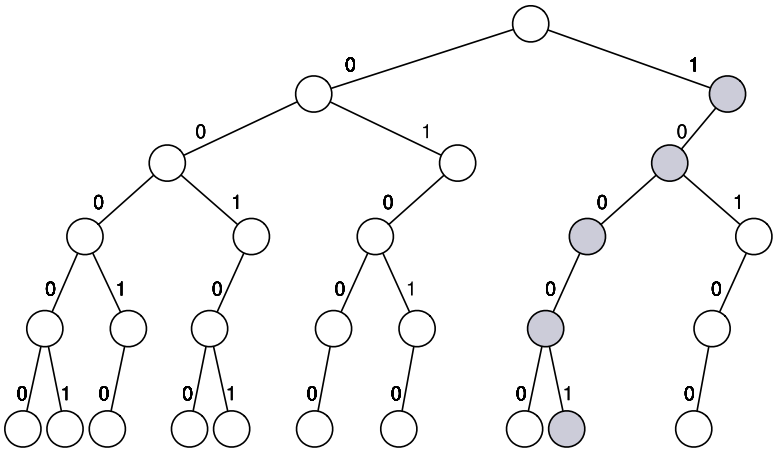


Figure 7.3: Creation of tree from subwords. Adapted from [42]. Note that the first subword from Figure 7.2 has been highlighted in the tree using darker nodes.

follow (or create if necessary) a branch to the left. If s is a 1, branch to the right. Continue similarly with the next symbol s in the subword. After branching is done on the final symbol in the subword, return to the root with the next subword. Thus, each path in the tree from root to leaf corresponds to a unique subword that has been seen so far.

3. **Build machine:** Find sets of "future-equivalent" nodes in the tree; these correspond to nodes in the reconstructed machine. Two nodes in the tree are future-equivalent if the branching structure of the subtrees rooted at those nodes are identical. Following Crutchfield and Young [25], Hanson defines an *L-morph* as a depth-L branching structure. Figure 7.4 shows the four distinct morphs present in the tree of Figure 7.3 (each labeled by a letter). As more data is obtained and the tree "fills out", the number of distinct morphs typically decreases.

   Each morph corresponds to a state of the epsilon machine. Each state thus represents sets of future-equivalent nodes. After the tree is labeled with the morph labels, transitions between states in the machine can be read off

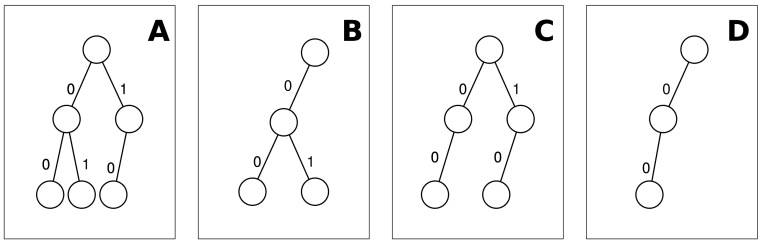


Figure 7.4: Different ($L = 2$) morphs contained in tree of Figure 7.3. Adapted from [42].

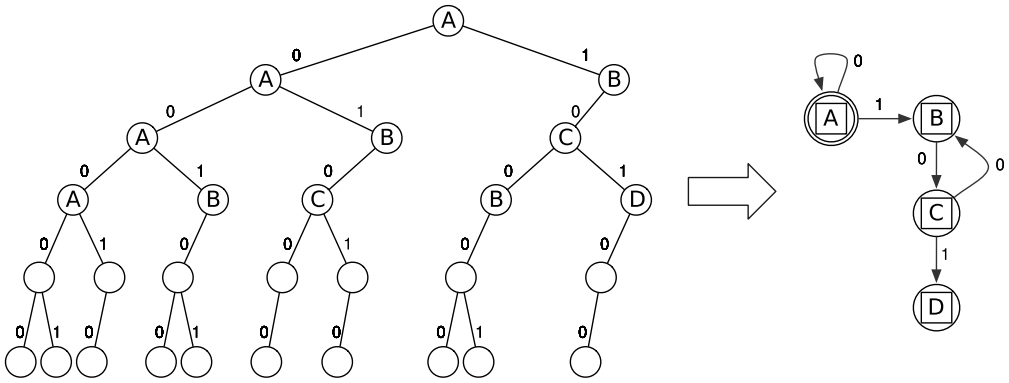


Figure 7.5: Left: The tree of Figure 7.3, labeled with morph labels. Right: The resulting epsilon machine. Adapted from [42].

the tree, as illustrated in Figure 7.5. The resulting epsilon machine can be tested to see if the regular language it recognizes fits the definition of a regular domain [42].

In [42], Hanson describes how to best choose the D and L parameters.

Given sufficient statistics, the epsilon machine resulting from this method can be said to capture the "intrinsic computation" being performed by the cellular automaton (or any system producing symbolic temporal dynamics). Such a machine can easily be used to create a regular-domain filter, like the ones used to produce Figures 6.4 and 6.5(b).

Shalizi [114] showed that epsilon-machines, if reconstructed accurately, are unique minimal representations of regular domains. Of course the reconstruction

method sketched above can reconstruct epsilon machines accurately only if given sufficient data. In [114] Shalizi discusses limitations of various epsilon-machine reconstruction methods, including the one sketched above.

The determination of regular domains (and thus particles) in cellular automata via the reconstruction of epsilon machines has some significant limitations. First, the accurate reconstruction of epsilon machines in order to determine regular domains can be highly computationally expensive, and can require significant amounts of data and correct determination of the parameters D and L. Second, it is not clear if all domain patterns of interest can be represented as *regular* domains (such as the complex domains pictured in Figure 6.2). Third, there are currently no general methods for representing higher-dimensional "domains" in terms of regular languages (Shalizi [114] discusses some ways in which such methods might be developed).

Other groups have proposed alternative methods for automatic discovery of computational structures in cellular automata (and in discrete space-time data in general) that address these limitations. That is, these methods do not assume any particular model (such as finite state automata) for representing patterns; they are easily extensible to two and higher dimensions, are less sensitive to particular parameter settings, and are computationally tractable. In the remainder of this paper we explore three such approaches: filtering by local sensitivity [115], filtering by local statistical complexity [115], and filtering by information storage, transfer, and modification [71]. First we will informally describe these methods, then we will qualitatively evaluate performance of the filters on one- and two-dimensional cellular automata. The filters are evaluated on their accuracy to outline the domain borders, and the computational requirements to do so. The formal definitions for each of the methods are included in the Appendix B, while the following sections describe these concepts in more intuitive terms.

7.2 FILTERING BY LOCAL SENSITIVITY (LS)

According to Das et al. [28], the computation of the density classification task is carried out by the particles. That means, the sites inside of the regular domains have different function than the ones forming particles. The local sensitivity (LS) filter has been proposed by Shalizi et al. to test if a site belongs to a structure[2] that will significantly influence future CA behavior [115]. This filter assigns to each cell c at each time-step of the CA space-time diagram a measure of cell c's sensitivity to perturbations. The larger the local sensitivity, the more likely it is that perturbations in or close to c will result in significantly different dynamics in sites that depend on the information stored in c. Conversely, when local sensitivity is small, it means that perturbations will often be "self-healed" by the dynamics of the CA, often leading to the stable behavior in the observed sites. Local sensitivity is inspired by the Lyapunov exponent of a dynamical system, which measures the rate of divergence between the trajectories starting from two infinitesimally close configurations of the system [60].

More concretely, let $\vec{\eta}$ denote the spatial coordinates of cell c and let t denote the time step at which a measurement is made. The pair $(\vec{\eta}, t)$ locates a specific site in a space-time diagram. Let p denote a perturbation radius around site $(\vec{\eta}, t)$ (where p is not necessarily equal to the CA rule radius r), and let P denote the set of sites contained within that perturbation radius, including the site $(\vec{\eta}, t)$ itself. The local sensitivity $\xi(\vec{\eta}, t)$ is calculated as follows:

1. Generate the set S that contains all the possible perturbations of sites in p. For example, for a binary-state CA with $P = 1$, suppose the states of cell c and its two nearest neighbors are (from left to right) 000. Then

[2]The semantic meaning of the structures and the particles is the same. The structures detected by the LS filter are the same as the particles found by the regular language filters.

$S = 001, 010, 011, 100, 101, 110, 111$. In general, the size of S is

$$|S| = k^{|p|} - 1,$$

where k is the number of cell states allowed by the CA.

2. Replace the states in the sites that correspond to the perturbation neighborhood in the current lattice with each $s \in S$ (keeping all the other sites in the lattice unchanged), and run the CA for d time-steps, where d is a *future-depth* parameter.

3. For each run of the CA for d time steps, record the states of each of the sites in the lattice, at each time $t' \in \{t+1, t+2, ..., t+d\}$, that depend on $(\vec{\eta}, t)$, and calculate the fraction of these that are different from the sites at the same positions within the original space-time diagram.

4. The local sensitivity $\xi_d^p(\vec{\eta}, t)$ is equal to the average of these fractions for a specific permutation range p and future depth d.

Determining the local sensitivity of some site $\vec{\eta}$ at some time t requires the choice of two parameter values, (1) the perturbation range p, and (2) the future-depth parameter d. As Shalizi et al. point out, the choice of these parameters is like adjusting a microscope, in order to make certain features of the observed object or phenomenon more salient at the expense of blurring others [115]. There is no clear heuristic to determine optimal values for these parameters.

Figure 7.6 illustrates the process of calculating the local sensitivity, $\xi_d^p(\vec{\eta}, t)$, using elementary CA 110 as an example. The original initial configuration is shown in the top-left corner, and the CA has been iterated for $d = 2$ time steps. The goal is to calculate the local sensitivity of the fourth site from the left, at time $t = 0$, i.e. $\xi_2^1(4, 0)$. This site is marked with a circle. The perturbation range is set to $p = 1$. In parts (A)–(G) of the figure, each perturbation s replaces the original

states of the three sites in P, and for each perturbation, the CA is iterated for $d = 2$ time steps.

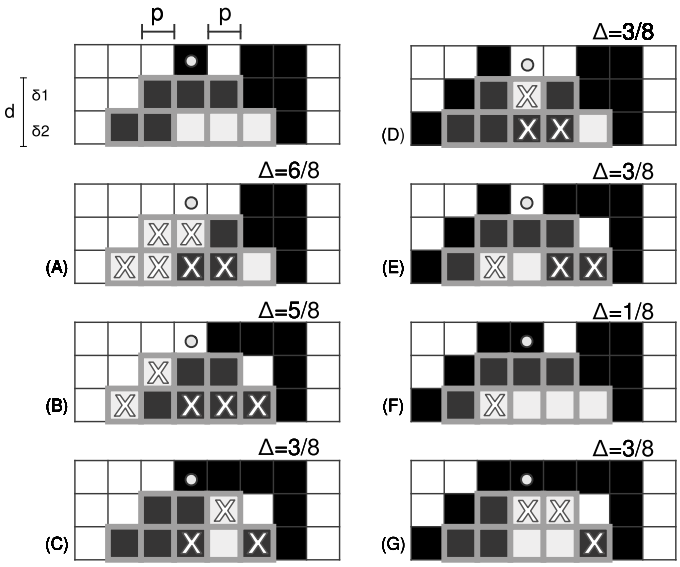


Figure 7.6: Example of the calculation of $\xi_2^1(4,0)$. The top-left corner shows the original initial configuration and two updates using elementary CA 110. The local sensitivity is calculated for the fourth site at time $t = 0$ (highlighted with a circle). The perturbation range is $p = 1$, which determines the perturbation neighborhood $P =< 0, 1, 0 >$, and the future depth is set to $d = 2$. The sites that depend on the information stored in site $(4, 0)$ (for a 1D radius $r = 1$ CA, within future-depth 2) are marked with grey squares. They determine the future light-cone for $(4, 0)$. The perturbation neighborhood P generates $|S| = 7$ "words" of length three. Diagrams (A)–(G) show the behavior of the CA when each of these words replaces the original configuration in the perturbation range, and the CA is run for $d = 2$ time steps. The cells in each future light-cone that differ from the corresponding cells in the original future light-cone are marked with an X. The Hamming distance Δ, i.e., the fraction of differing cells between the original future light-cone and the one resulting from each perturbation is shown above the top-right of each diagram in (A)–(G). These seven Δ's are averaged, resulting in a local sensitivity $\xi_2^1(4, 0) = 0.4285$.

In each diagram in Figure 7.6, the *future light-cone*[3] $l^{+}_{(d=2)}(4,0)$ of the site (4,0) is highlighted with gray outline. The future light cone of a site $(\vec{\eta}, t)$ (to depth d) is the set of sites in the next d time steps that are influenced by information stored in site $(\vec{\eta}, t)$. Clearly a perturbation of a state in site $(\vec{\eta}, t)$ can affect the states of sites only within that site's future light-cone.

Note that the future light-cone of a site is determined by the topology and radius of the CA rule, not by the perturbation neighborhood P. In the example illustrated in Figure 7.6, the rule's local neighborhood and perturbation neighborhood radii are the same, i.e. $p = r = 1$; however, the choice of the perturbation range is not constrained by the radius of the CA rule.

For each future light-cone resulting from a perturbed configuration, each of the light-cone's sites is compared with the corresponding site in the original $l^{+}(4,0)$. The fractional Hamming distance[4] between the original and perturbed light-cones is the fraction of sites that differ (i.e., those marked by Xs in Figure 7.6). The local sensitivity[5] $\xi^{1}_{2}(4,0)$ is simply the average of these fractional Hamming distances. In the example shown in Figure 7.6,

$$\xi^{1}_{2}(4,0) = 0.4285.$$

7.3 FILTERING BY LOCAL STATISTICAL COMPLEXITY

The local statistical complexity (LSC) filter was also proposed by Shalizi et al. [115] as a computationally cheaper filtering alternative. The execution times of the LS filter are high because at each site the LS computes and analyzes the future light-cones for all possible permutations of the initial configuration of sites with a given radius. The LSC on the other hand, needs only two passes through a single

[3] *see Appendix, Definition B.1.1 - B.1.5*
[4] see Appendix, Definition B.1.6
[5] see Appendix, Definition B.1.7

space-time diagram. First, it collects statistics about past and future behavior at each site; during the second pass it assigns the complexity values.

Like local sensitivity, LSC is a value computed at each site in a space-time diagram. Figures 7.7 and 7.8 illustrate the process of computing the LSC value at a given site. The site under consideration is marked with a circle, and its past and future light cones[6], each to depth $d = 2$, are outlined in gray. Denote the depth-d past light-cone associated with a site c as $l_d^-(c)$ and the corresponding future light-cone as $l_d^+(c)$. Note that the same past-future-light-cone pair might appear elsewhere in the space-time diagram, and also that a pair $(l_d^-(c),\ l_d^+(c))$ does not include the cell c itself.

In computing LSC, we will count the number of times each possible past/future pair appears in the diagram. In general, assuming a binary-state CA and depth 2 light cones, there are 2^{16} possible pairs (2^8 for each 8-cell past light-cone and 2^8 for each 8-cell future light-cone).

Each site's past and future light-cones are extracted, and added into a conditional distribution matrix[7] M as follows: If the past light-cone l^- or the future light-cone l^+ has not been seen before, add a new row or column to M, the index of which, i (for past) or j (for future) represents the newly observed light-cone, and assign $m_{i,j} = 1$, that is, the past light-cone configuration i has been seen *once* leading to the future light-cone j. If the past (or future) light-cones have been seen before, then the value of $m_{i,j}$ is increased by one. This yields a matrix that contains the conditional frequency distributions of every distinct past history over the future light-cones that have been seen in the specific CA space-time diagram.

The notation $\vec{m}_i^+$ will be used to denote a row[8] in M, corresponding to the conditional distribution of futures given the past light-cone represented by the i^{th}

[6] see Appendix, Definition B.2.1
[7] see Appendix, Definition B.2.2
[8] see Appendix, Definition B.2.3

row of M, i.e. $P(l^+|l_i^-)$. Shalizi et al. point out that if two rows $\vec{m}_a^+$ and $\vec{m}_b^+$ are equal, then the two past light-cones are *equivalent* [115]. The *equivalence classes* that result from clustering the observed past light-cone configurations are called *causal states*[9], where each of these contains a number of equivalent past light-cones: those that predict the same possible futures with the same probabilities[10]. Each site of the space-time diagram is assigned the local statistical complexity[11] value of the corresponding casual state that the site's past light-cone belongs to.

Before discussing more specifically how LSC is computed for each site, the following point is important to note. For the LSC filter, since it is possible to gather only finite samples, and due to the practical necessity of limiting the depth of recorded past and future light-cones, the statistics that are used to estimate causal states will always have an error margin; that is, the conditional distributions just described have to be estimated from data. Using the estimated conditional distributions $P(l^-|l_j^+)$ computed from at least one space-time diagram using a past/future depth parameter d, it is often possible to produce reasonable approximations to the true set of causal states. These approximations of causal states are computed by clustering past light-cone configurations that have a *statistically similar*[12]. distribution over future light-cones.

The procedure for computing LSC for each site is as follows. Once $\mathbf{M}$ has been filled in, the next step is a traversal of all the rows in $\mathbf{M}$, with the goal of assigning each row, m_i, to a *causal state*. Before the traversal, the procedure starts with an empty set of causal states ϵ, and marks every row in $\mathbf{M}$ as *unassigned*. The procedure then (1) finds the next unassigned row m_i in $\mathbf{M}$, and makes it its own causal state, ϵ_i; (2) computes the similarity between m_i and every other unassigned row m_j. If the similarity is above a threshold, then m_j is added to the same causal

[9] see Appendix, Definition B.2.5
[10] see Appendix, Definition B.2.7
[11] see Appendix, Definition B.2.8
[12] *See Appendix, Definition B.2.4*

state as m_i; if not, m_j remains unassigned. The procedure uses a χ^2 test to measure whether frequency distributions m_i and m_j are significantly similar (above a given threshold). This process is repeated until all rows are assigned to a causal state.

After this part of the procedure is complete, it is then necessary to calculate the probability that a past/future light-cone pair is in a specific causal state. Note that the total number of observations of l^-, l^+ pairs is $||M|| = \sum_{i,j} m_{ij}$. Similarly the total number of observations associated with causal state ϵ_i is $||\epsilon_i|| = \sum_{i,j|m_{ij} \in \epsilon_i} m_{ij}$. The probability that a past/future light-cone pair is in causal state ϵ_i is therefore, $Pr(\epsilon_i) = ||\epsilon_i||/||M||$. Finally, the statistical complexity of a site c is given by $C(c) = log_2(Pr(\epsilon_i))$, where ϵ_i is the causal state to which belongs the combination of past and future light-cones associated with site c

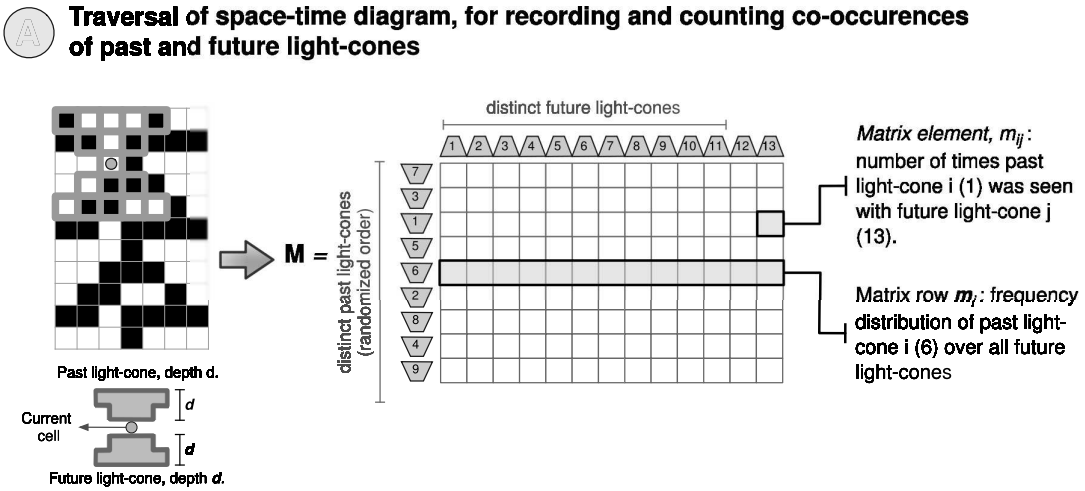


Figure 7.7: The first part of the procedure to compute Shalizi's *Local Statistical Complexity* consists of traversing a space-time diagram to gather statistics about the set of unique past and future light-cones observed. Each unique past (future) light-cone is assigned an index, i (j). The set of all past (future) light-cones is denoted by P_c (F_c). For every site that has a past and future light-cone, first identify past and future light-cones i and j. Then in a matrix $M_{|P_c|\times|F_c|}$ (where initially $m_{ij} = 0, \forall i, j$), increment the value of m_{ij} by one. In the figure there are nine and thirteen *abstract* past and future light-cones respectively. All the elements of this figure are only simplified illustrations of the concepts introduced. Note that a row $\mathbf{m_i}$ in M represents the frequency distribution of past light-cone i over all the future light-cones. Finally, after the M has been updated upon traversal of the space-time diagram, the ordering of its rows is *randomized* (see justification for this in the text).

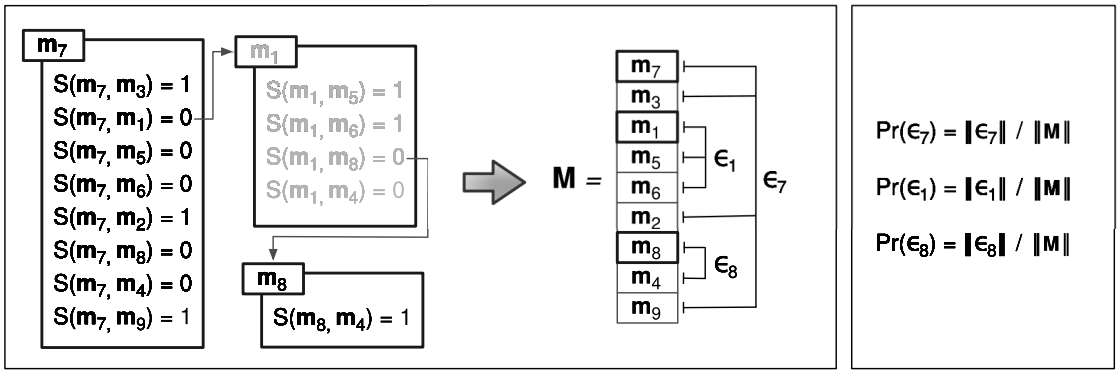


Figure 7.8: The procedure for computing LSC continues with a traversal of all the rows in M, with the goal of assigning each row, m_i, to a *causal state*. Before the traversal, start with an empty set of causal states ϵ, and mark every row in M as *unassigned*. In this example, the procedure finds the first unassigned row m_7 in M, and makes it its first causal state, ϵ_7; then it computes the similarity between m_7, and every other unassigned row: If the similarity $S(m_7, m_j) = 1$, then the row is added to the same causal state as m_7, if not, the row remains unassigned. Here, rows m_3, m_2, and m_9 are statistically similar to the ϵ_7 representative row m_7. The next unassigned row m_1 is chosen to represent new causal state ϵ_1. The similarity calculation adds rows m_5 and m_6 to the causal state ϵ_1 (column 2). The same procedure is repeated for the reminding unassigned row m_8. After this part of the procedure is complete, it is then necessary to calculate the probability that a past/future light-cone pair is in a specific causal state. In the figure, the total number of observations is the city-block norm of $||M||$, $||M|| = \sum_{i,j} m_{ij}$. Similarly, the total number of observations associated to a causal state ϵ_i is $||\epsilon_i|| = \sum_{i,j|m_{ij} \in \epsilon_i} m_{ij}$. The probability that a past/future light-cone pair is in causal state ϵ_i is therefore, $Pr(\epsilon_i) = ||\epsilon_i||/||M||$. This calculation for causal states ϵ_7, ϵ_1, and ϵ_8 is shown in the far right column. Finally, the statistical complexity of a site c is given by $C(c) = log_2(Pr(\epsilon_i))$, where ϵ_i is the causal state to which the combination of past and future light-cones associated with site c belongs to. Note that the assignments made in the figure are only an abstract illustration of the procedure.

The exact number of causal states, and which matrix row best represents each causal state, could easily be determined from an **M** matrix in which the conditional probabilities had converged (given an infinite number of encountered configurations). In practice, only a finite set of sites is analyzed, which causes the number and definition of a causal state to depend on the order in which rows are retrieved from **M**. To remedy this bias, the matrix rows are retrieved from **M** at random [116]. Each retrieved row is either assigned to an existing causal state, or such row represents a newly created causal state.

Although the underlying mechanics of the LSC filter are different from the computational mechanics framework introduced in §7.1, the *L-morphs* used in the former and the *causal states* just introduced are conceptually equivalent: both these characterizations represent sets of past histories of a dynamical system that predict the same futures, with the same probability distributions. This makes it possible to cluster potentially large numbers of a system's histories as one of a few causal states, and thus reveal the higher-level coherent structures formed in its dynamics, as well as the interactions among such structures.

7.4 FILTERING BY INFORMATION STORAGE, TRANSFER AND MODIFICATION

The LS and LSC filters described above use the entire past and future light-cones of a given site to measure the degree of "information processing" occurring at the given site. Lizier, Prokopenko, and Zomaya [71] proposed an alternate set of filters for detecting information processing at a given site, which use only a limited range of past site configurations. Their three proposed filters are meant to measure three aspects of information processing at a given site: local information storage, transfer, and modification.

7.4.1 Local Information Storage (IS)

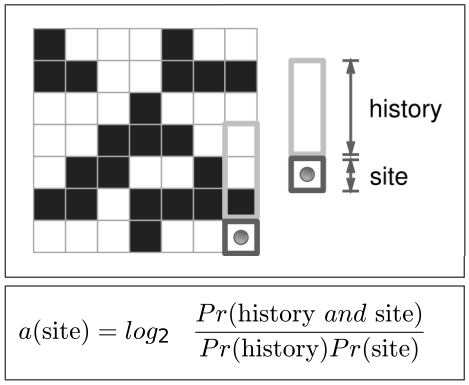


Figure 7.9: Pictorial description of the procedure to compute Lizier et al.'s local information storage $a(site)$, in this case, with $k = 3$. The site for which a is being computed is outlined and marked with a circle. The three sites forming the site's "history" are outlined in light gray. The actual computation is described in the text.

Lizier et al.'s *local information storage* filter (referred to in their paper [71] as "local active information storage") measures the degree to which the state at a given site s, at a given time t, is predictable from previous states at that same site. In other words, it measures the degree to which the current state "remembers" (has mutual information with) its direct history.

Consider the simple example of a site s that is in the black state for k time steps, but at time step t changes to the white state. If such a change is seen only rarely over the CA lattice, the site at time t would be considered to have low information storage, since the statistics of past consecutive black states would "misinform" us about the current state. In contrast, consider a CA which continually alternates between all black and all white states. Every site would have high information storage—the statistics of past states give us perfect information about the current state. Thus, sites inside regular domains, as described in previous sections, would have relatively high information storage.

Define $x_{i,n}^{(k)}$ as the vector of the past[13] k states of site i ending with time step n, and define $x_{i,n+1}$ as the state of site i at time step $n+1$. For example, let $k = 3$ and consider the (small) space-time diagram of Figure 7.9. Let $i = 7$ (i.e., the rightmost site in the lattice) and $n = 6$ (i.e., we are calculating information storage for the site at time step $n+1 = 7$. Then $x_{i,n+1}$ is **white** (the site is marked with a circle), and $x_{i,n}^{(k)}$ is (**white, white, black**).

The local information storage[14] of site i at time step $n+1$ is defined by Lizier et al. as the base-2 logarithm of the probability that history[15] $x_{i,n}^{(k)}$ will be followed by state $x_{i,n+1}$ at site i. This is computed by examining, over N space-time diagrams, all single-site histories of length $k+1$, and over these counting the number of times the particular history $x_{i,n}^{(k)}$ is followed by the particular state $x_{i,n+1}$, as well as counting their independent occurrences. (To be precise, information storage is defined by Lizier et al. as the limit as k approaches infinity of this logarithm; see Appendix, Definition B.3.3. In practice, information storage is estimated with a finite value of k.)

In the very simple example given in Figure 7.9, we first ask how many three-site histories there are in the diagram, where a three-site history is a vertical line of three sites, followed by a fourth site. We can see that such histories occur starting only in the first four rows, since the sites in the fifth, sixth, and seventh rows don't have enough (vertical) data to start such histories. Thus there are $4 \times 7 = 28$ such histories. Of these, we count six histories with the pattern "white, white, black". Similarly, there are 28 individual sites that follow (in a vertical column) a three-site history—i.e., the sites in the fourth through seventh rows—and of these, 15 are white. Finally, we ask, how many of the three-site histories of "white, white, black" are followed by a white site? Here we count 2—i.e., two of the "white,

[13] See Appendix, Definition B.3.1.
[14] See Appendix, Definition B.3.3.
[15] See Appendix, Definition B.3.2.

white, black" histories are followed by a white site. Thus, the information storage a of the given site is:

$$a(site) = log_2\left(\frac{Pr(history \text{ and } site)}{Pr(history)Pr(site)}\right) = \frac{2/28}{(6/28)(15/28)} = -0.68$$

This negative value indicates that the site in question has low information storage, in that its white state is not well-predicted by its "white, white, black" history, since most of the "white, white, black" histories in the diagram are followed by a black state.

In practice, when calculating this statistic, much larger space-time diagrams and larger values of k would be used, to get better statistics. Intuitively, as $k \to \infty$, the statistics about a site's history are more complete and the filter's accuracy increases. The calculation would involve sites that occur only after initial transients have died out.

7.4.2 Local Information Transfer (IT)

The IS filter takes into account how well the configuration of a site's history predicts the state of that site, but it does not take into account values from any other spatial locations. However, a site's next state is directly affected by the site's r adjacent neighbors. The local information transfer (IT) filter takes these neighbors into account in order to measure the information *transfered* into a given site.

The left information transfer t_{left} and the right information transfer t_{right} filters measure how much information was contributed —i.e., *transferred in space*—by the left or the right neighbors to the predictability of the current site's value. The t_{left} filter is defined as the base-2 logarithm of the conditional probability of the site's state with respect to both the site's length-k history and its left neighbors (as defined by the CA's radius r), divided by the conditional probability of the site with respect to its history alone. Intuitively, this filter gives the amount of information

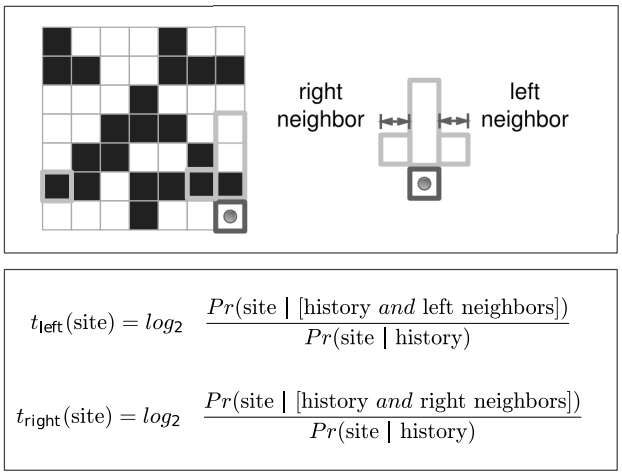


Figure 7.10: Pictorial Description of the procedure to compute Lizier et al.'s right and left information transfer. The site for which information-transfer is being calculated is marked with a gray circle. Its three-site history is outlined in light gray, as are the left and right neighbors at $t = 6$. Note that, due to the circular boundary conditions, the "right neighbor" is actually the leftmost site at $t = 6$. Details of the calculation of Left and Right Information transfer are described in the text.

about the site in question that is provided by the *transfer* of information from its left-neighbor in the previous time step, apart from any information about the site already contained in its history. The t_{right} filter is defined analogously. Thus, sites that are part of *left-moving particles* would have high left-information transfer values, and sites that are part of *right-moving particles* would have high right-information transfer values.

Figure 7.10 gives a pictorial illustration of how these values are calculated. As in Figure 7.9, the site in question is marked with a circle, and $k = 3$. Here the neighborhood radius r is equal to 1. To calculate t_{left}, we count the fraction of occurrences of the "white, white, black" history alone (6/28) and followed by a white site (2/28), the fraction of joint occurrences of the"white, white, black" history and black left-neighbor at $t = n$ (2/28), and the fraction of joint occurrences of the "white, black, black" history, the black left-neighbor, and the white site itself

(2/28). We thus have

$$\begin{aligned} t_{\text{left}}(site) &= log_2\left(\frac{Pr(site|history \text{ and } left\ neighbors)}{Pr(site|history)}\right) \\ &= log_2\left(\frac{(2/28)/(2/28)}{(2/28)/(6/28)}\right) \\ &= 1.6. \end{aligned} \tag{7.1}$$

This positive value for t_{left} makes sense since in this example the site's state is well predicted by the state of its left-neighbor at the previous time step. In other words, positive information about the state of the left neighbor at the previous time step can be said to have been *transferred* to the site in question.

Noting that that due to the circular boundary conditions, the "right neighbor" of the site in question is actually the leftmost site, t_{right} is calculated as follows:

$$\begin{aligned} t_{\text{right}}(site) &= log_2\left(\frac{Pr(site|history \text{ and } right\ neighbors)}{Pr(site|history)}\right) \\ &= log_2\left(\frac{(2/28)/(4/28)}{(2/28)/(6/28)}\right) \\ &= 0.58. \end{aligned} \tag{7.2}$$

This value is positive, meaning that the right neighbor's state has some predictive information for the site in question, though it is not as predictive as the left neighbor. Thus one can say that positive information was transferred from the right, though not as much as was transferred from the left.

As was the case for information storage, the precise definition of information transfer requires taking the limit as k approaches infinity of the base-2 log of the conditional probabilities. However, in practice it is estimated with finite values of k.

7.4.3 Local Separable Information (S) and Information Modification (IM)

The local separable information filter (S) measures, at a given site, the extent to which the site's state is predicted well by its history and/or neighboring states. That is, the sites at which both information storage and information transfer are low are said to have high *local information separation* s[16]. A site with negative local information separation is said to be a site at which information (from the site's history and/or neighbors) has been *modified*. Lizier et al. interpret such sites as the loci of information processing—e.g., where two particles collide and create a new particle.

Lizier et al. defined local separable information s as the sum of information storage and information transfer from all directions:

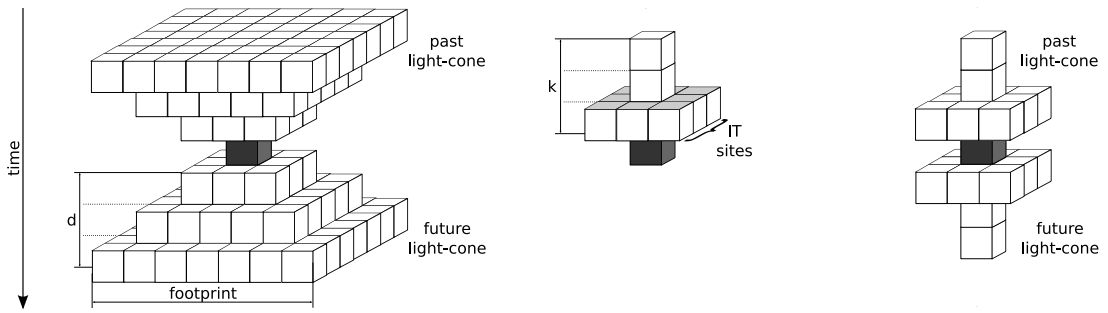
$$s(i,n) = [a(i,n) + t_{\text{left}}(i,n) + t_{\text{right}}(i,n)] \tag{7.3}$$

For the site illustrated in Figures 7.9 and 7.10, the value of s is calculated as: $s = -0.68 + 1.6 + 0.58 = 1.5$. This positive value indicates that the information at this site is "separable" (i.e., can be understood in terms of the history and/or neighbors) and is thus not a locus of "information modification".

7.5 FILTERING COHERENT STRUCTURES IN TWO-DIMENSIONS

Crutchfield and Hanson's method (see Section 7.1) relies on processing the sub-words that make up the pattern of domains in 1DCA in order to build an epsilon machine [23, 25, 44]. The resulting finite-state automaton is then used to filter out the input patterns that belong to domains. Unfortunately, the authors did not extend their filtering approach or grammatical inference techniques into two

[16]See Appendix, Definition B.3.5.



(a) Shalizi et al. (b) Lizier et al. (c) Hybrid

Figure 7.11: Space-time patterns used for gathering statistics in (a) Shalizi et al.'s local sensitivity and local statistical complexity filters (b) Lizier et al.'s information theoretic filters, and (c) our hybrid filters. The patterns here have two spatial dimensions (corresponding to the CA lattice) and one temporal dimension. Filters proposed by Shalizi et al. use light-cones with depth d with cone width marked as $footprint$. Lizier et al.'s IS filter uses site's past configurations k tall, and the IT filters use additional information from 3×3 sites in a given site's neighborhood. The hybrid filter uses Lizier et al.'s past and future light-cone configurations.

dimensions. Doing so would present a number of difficulties, including extending the notion of *regular language* to two dimensions [69] and significantly extending the *epsilon-machine reconstruction* method to overcome the structural and combinatorial differences between one- and two-dimensional pattern analysis.

The shortfalls of regular-language-based filters can be overcome by statistically-based filters. As one of the contributions of this thesis, I extended the Shalizi et al. and Lizier et al. one-dimensional filters into two-dimensions and I defined a novel two-dimensional hybrid filter that combines the two-dimensional filters into one.

Extending the LS and LC filters into two-dimensions requires collecting statistics about site configurations in three dimensions. The past and future light-cone configurations for Shalizi's 2D filters have a 3D pyramid shape (Figure 7.11 a). The LS algorithm is adapted to calculate Hamming distances on 2D slices of the 3D future cones, and the LC filter records the frequency distributions of the 3D cones for each site in the CA space-time diagram. Similarly, extending the IS, IT, and

IM filters for 2DCA requires recording structures in three dimensions (Figure 7.11 b). The information transfer for Moore neighborhood 2DCA is defined in eight directions, and a Cumulative Information Transfer (CIT) measure is defined as a sum of all directional IT filters. Other than accounting for 3D shapes of recorded structures and making minor algorithmic changes to account for these shapes, the methodology of calculating filter values remains the same.

The hybrid filtering approach combines Shalizi et al.'s algorithmic approach with Lizier et al.'s structures as following: the hybrid filter used the LC algorithm to record the frequency statistics of past and future configurations, to cluster frequency distributions with similar future behavior, and to calculate a site's complexity according to its cluster membership. Unlike the pyramid-shaped light-cones used by the original LC algorithm, however, the hybrid filter uses a cross-like 3D configuration proposed by Lizier et al. (Figure 7.11 (c)). This approach combines the strength of the LC filter to group frequency distributions with similar future configurations into behavioral classes which abstract common patterns as domains, and the narrow structures used by Lizier et al. which make the number of patterns tractable and outlines the domain borders with narrow lines. Small number of unique past and future 3D configurations made the hybrid filter scalable to large lattice sizes (up to 99×99).

7.6 RESULTS

The goal of filtering is to identify the coherent patterns that define domains, subtract these patterns from the space-time diagram, and highlight the borders between the domains. Accurate identification of the domain borders is crucial for building a model that captures the dynamics of the domain borders in space and time. If the model, whose primitive elements are particles rather than detailed CA configurations, accurately predicts a CA's performance on a task, we know that the filters have correctly identified the information-carrying sites. This demonstration

has been carried out for the one-dimensional density classification task [52] but such a model has not yet been constructed for the two-dimensional version.

In the absence, as yet, of such a model, here we present a qualitative evaluation of filtering methods. To evaluate each filters' performance we measure the accuracy with which the filter's output matches an "ideal" set of hand-constructed domain borders, the filter's capacity to identify small structures (single cell wide domains), and the filter's computational requirements and scalability to large lattices.

The two-dimensional versions of the Local Sensitivity, Local Complexity, Information Storage, Information Transfer, Information Modification, and Hybrid filters were applied on the space-time diagrams of the two-dimensional density classification task (Figure 7.12). The tested CA rules were evolved by genetic algorithms by Cenek [14], Marques-Pita [14], and Wolz and de Oliveira [14, 140] (see Chapter 3 for more details and Appendix A for bit-string representations of these rules). The experimental setup used a randomly initialized CA lattice of 39×39 cells that was updated 55 times. All three rules used the same random initial configuration, and the algorithms described above were implemented sequentially without any parallelization.

As a side note, Figure 7.12 presents the results of IT filter instead of the IS or IM filter results. This is because the quality of the cumulative information-transfer filter results could not be improved by the addition of the IS filter results. The shortcomings of the IS filter are discussed in Section 7.6.2.

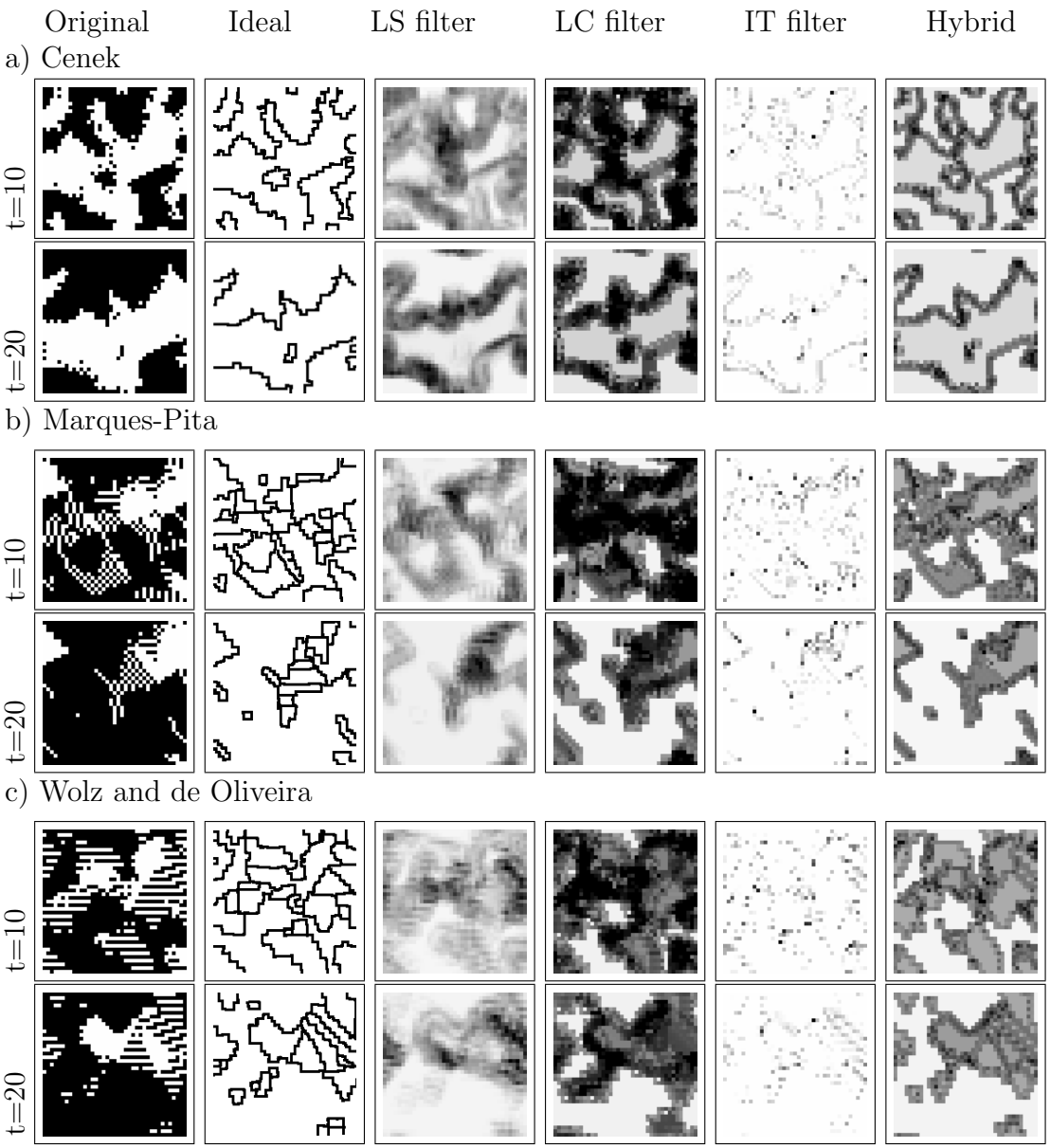
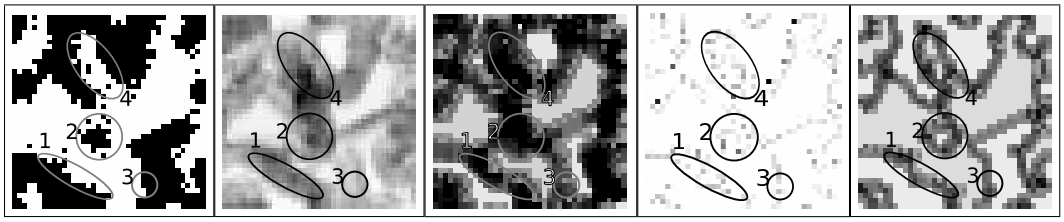


Figure 7.12: The results of the filters for (a) Cenek's rule, (b) Marques-Pita's rule, and (c) Wolz and de Oliveira's rule, each on a 39×39-cell lattice at time steps $t = 10$ and $t = 20$. The first column shows the original space-time diagrams, followed by the idealized domain-boundary outline (hand-constructed), filter results for local sensitivity (column 3), local statistical complexity (column 4), cumulative information-transfer (column 5), and the hybrid filter (column 6). The gray-scale in the images corresponds roughly to the likelihood that a given site belongs to a domain boundary—dark colors mean high certainty while light-gray sites are less likely to form a boundary. For additional results see [14].



(a) original CA (b) LS filter (c) LC filter (d) IT filter (e) hybrid filter

Figure 7.13: Cenek's rule for the density classification task on a 39×39-cell lattice at time step t=10. (a) The original CA, (b) the results of the local sensitivity filter, (c) the local complexity filter, (d) the cumulative information-transfer filter, and (e) the hybrid filter. Highlighted features represent the various filters' results on (1) a noisy border, (2) a small feature, (3) a zero-velocity border, and (4) a region with complex border dynamics (4).

7.6.1 Computational Requirements

The filters' computational requirements were evaluated in terms of run time and memory requirements. All experiments were run on Intel(R) Xeon(R) E5410 2.33GHz CPU with 4GB RAM. The algorithms were implemented sequentially without parallelism.

At each site of the space-time diagram, the LS filter analyzed future light-cones with depth $d = 5$ for all possible permutations of the initial configuration within the site's perturbation radius. A deeper light-cone depth caused an exponential increase in execution time and a decrease in accuracy to outline domain borders (discussed later in this chapter), while a shallower light-cone depth was insufficient for the LS filter to distinguish between sites with high and low local sensitivity. The light-cone depth of $d = 5$ was chosen as a compromise between the quality of the outlined domain borders and the algorithm's computational requirements. The execution time of the filter was approximately 14 hours.

The LC filter logs the occurrences of all unique past and future light-cones into a frequency matrix M. The GA-evolved 2DCA rules can produce ambiguous domain borders and make the domain collisions appear "noisy". The noise and

complex domain interactions caused the number of unique light-cone configurations to reach $19,000 \times 19,000$ for light-cone depth $d = 2$. The light-cone depth greater than $d = 2$ caused number of unique light-cone configuration to grow so large that the algorithm's execution become infeasible. The number of unique light-cones did not asymptotically decrease with time, and new configurations were recorded even at time $t = 55$. Given these characteristics, the time and space complexity of the LC filter became prohibitive.

Although the IT filter collects statistics from multiple randomly initialized CA lattices before analyzing a given CA, this filter is computationally less expensive. The execution time of a hybrid filter was approximately 20 minutes, and $2,000$ unique 3D cross-like configurations (Figure 7.11b) were detected.

The hybrid filter uses the same 3D configurations as proposed by Lizier et al. which limits the number of unique past and future configurations. The hybrid filter uses the original LC algorithm, as proposed by Shalizi et al., to record frequencies of 3D cross-like configurations into matrix M, perform clustering of the past frequency distributions, calculate the clusters' complexity, and assign an LC value to each site. The execution time of the hybrid filter was approximately 2 hours and the number of unique past and future configurations was around $2,000$.

7.6.2 Results of Filtering

Figure 7.12 presents a side-by-side qualitative comparison of filtering results. The second column shows the output of an ideal filter, obtained by hand-segmenting the original space-time diagram to determine the exact location of the domain borders. In general it is not trivial or definite to decide on the location of the domain borders. For an automated approach to achieve a comparable quality of results to the ideal filter, it must take the binary image created by a CA and outline the domain border by a continuous, single-cell wide edge, disregarding the noise around the borders, and making a clean, single-cell wide border where two

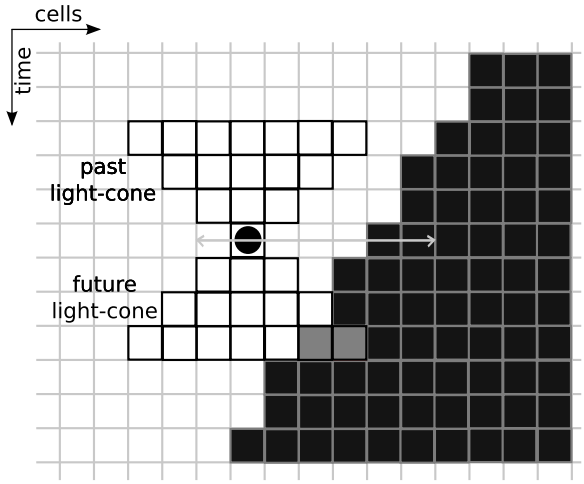


Figure 7.14: A two-dimensional cut of a 3D space-time diagram along the time axis shows a current site as a black circle, along with the time-slice of its past and future light-cones. A black domain is shown on the right side of the illustration. The gray colored sites show an intersection between the future light-cone and the black domain. The gray arrow shows the range of sites that the LS and LC filters highlight as a wide blurry border between the white and black domains.

or more domains collide. It must also highlight small and single-cell wide domains with a narrow edge.

We compared the output quality of the LC, LS, IT and Hybrid filters with respect to the ideal segmentation. The LS and LC filters highlight domain borders as wide and blurry bands (Figure 7.14). This is because both filters use pyramid shaped structures with a wide footprint at depth d. For example, consider a site near a zero-velocity domain border. The LS filter will perturb the initial configuration within the perturbation radius and update the sites affected by this perturbation—a pyramid shape structure. The current site is assigned a non-zero LS value, despite the fact that neither the original site nor the sites in its perturbation radius form the domain border, because the light-cone of the affected sites intersect with the neighboring domain. Analogously, the LC filter will also assign the current site a non-zero value. This is because the site's future light-cone crosses over to the neighboring domain, so the site's future configuration

is that of a border region rather than belonging to a domain. The width of the border increases for the domains with non-zero velocity. Features 2 and 4 in Figure 7.13 (b,c) illustrate the filters' failure to detect a small domain and a region with complex dynamics. The output of the LS and LC filters lacks detail because the domain borders are highlighted as wide and blurry bands.

The IT filter proposed by Lizier et al. outlines the domain borders with a narrow line, but fails to detect vertical particles that in 2DCA are represented as domain borders that do not move (Figure 7.13, feature 3). At such places, the filter's output results in boundary discontinuities. Using the IT filter, the location of a border is also unclear around regions with noisy and complex behavior (Figure 7.13, features 1 and 4, respectively).

Qualitatively, the best filtering results were achieved when the approaches of Shalizi et al. and Lizier et al. were combined. Figure 7.12, column 6 shows the results of the hybrid filter. The domain borders produced by the filter are relatively narrow in comparison to LS and LC filters; for example, features 2 and 4 in Figure 7.13 (e) were highlighted without loss of detail. The filter also accurately highlighted a section of a domain border with zero-velocity shown as feature 3 in Figure 7.13e.

The proposed goals for filtering methods (to identify coherent patterns as domains, to outline them with narrow and accurate borders, and to compute the results in minimal time and memory) were best met by the hybrid filtering method. The filters proposed by Lizier et al. have qualitatively weaker results, and need to collect statistics from multiple space-time diagrams. Finally, Shalizi et al.'s LS and LC filters were computationally the most expensive, and the quality of results was the worst.

7.7 DISCUSSION AND SUMMARY

To start the discussion, let's pose several questions first: Do the statistical filters reveal anything about the nature of collective computation in the CAs? What did we gain by using statistical-based filters? Is there an alternative approach to statistical data-driven filters that would characterize the collective computation in the CA lattice?

First, let's examine how Crutchfield et al. built and validated the computational mechanics framework for 1DCA [23, 25, 44]. The framework can be summarized as a two step process. First, Section 7.1 describes the process of epsilon machine reconstruction – a regular-grammar-based filtering technique – to highlight sites in the CA lattice that seem to be of importance to characterize the computation in the lattice. At this point, it is only a hypothesis that the highlighted regions (later referred to as particles) are responsible for the mechanism of computation in the lattice. The second step requires building a model that captures the movement and interaction of particles over time (also called a dynamic model). After the first several time-steps, the state of a lattice can be described by the regular domains and the particles that separate these regions. At this point, instead of using the look-up table to calculate the next time-step, the previously detected particles are substituted into the lattice as vectors. Each particle is described by velocity, direction, a position in the lattice, and the interaction outcomes with other particles. The time and location of the first particle collision is calculated using vector physics. The particles at the collision site are updated based on the particles' interaction table. The position of the remaining non-collision particles is updated for the current time-step. The process of calculating the next particle interaction is repeated until the lattice contains no more particles or no further collisions can be calculated (parallel particles) [50]. Hordijk et al. confirmed that such a particle-level description of a CA's dynamics captures the mechanism of

collective computation in the 1DCA [52]. The model was validated by comparing the behavior of several evolved CA rules each on $10,000$ randomly initialized ICs, with the behavior of each rule's computational model. Since the model correctly predicted the CA behavior, the original hypothesis that the patterns of highlighted sites (particles) capture the mechanism of collective computation in the evolved CAs was confirmed (for more details see [50, 52]).

This chapter compared the ability of several filtering methods to detect the domains and highlight the sites that seem to be of importance for the mechanism of collective computation in 1DCA and 2DCA. As of yet, there is no model to simulate particle interactions in two dimensions, so it is currently impossible to validate that the patterns and the dynamics of these sites predict the performance of the CA lattice.

The claim by Lizier et al. that "the local transfer entropy provided the first quantitative support for the long-held conjecture that particles are the information transfer agents in CAs" [71] is unfounded. The filters' output serves only as an interpretation of the local site's "informational significance." The filters' output alone cannot predict the long-term behavior of a non-linear system; the dynamics of lattice-wide patterns (beyond a site's neighborhood radius) highlighted by the filters is what gives rise to the emergent computation in the lattice. This is the reason why a conclusion is premature, and we can only hypothesize if the highlighted sites correctly characterize the mechanism of collective computation.

It appears that the statistically based filters applied to a one-dimensional density classification task on Das et al.'s rule ϕ_{par} detect the same particles as the particles detected by the regular grammar based filters [76]. Although these particles were validated by Hordijk el. al.'s dynamic model as information carrying structures, the similarity between the particles highlighted by the statistically based and lambda reconstruction filters does not imply that the statistically based filters correctly detect the structures that capture the mechanism of collective computation

in the CAs.

Using the statistically based filters allows for the detection of coherent spatio-temporal patterns in higher dimensions, the abstraction of domains that consist of non-trivial patterns undetectable by a regular grammar, and the conceptualization of domains and domain borders for rules with noise. The filters proposed by Shalizi et al. and Lizier et al. suffer from several drawbacks. These include Shalizi et al.'s poor resolution of highlighted border regions, and infeasible computational requirements due to the large number of unique light-cone configurations. In addition, Lizier et al.'s filters are unable to detect zero-velocity domain borders or small domains. The best results in terms of the accuracy in highlighting domain borders and of the computational requirements were achieved by combining these two filtering methods, creating a hybrid filter.

Chapter 8

DYNAMIC MODEL

The previous Chapter defined and tested statistically based filtering techniques as the first step towards understanding the nature of emergent computation in a two-dimensional cellular automaton. The filters identified the coherent spatio-temporal structures with information content. The coherent domain patterns were identified by the filters as information storing or having low information complexity. The domains were "subtracted" from the lattice, and the domain borders were revealed as the information carrying structures. The hypotheses that the structures highlighted by the filters accurately describe the mechanisms of computation must be confirmed.

One way to validate this hypothesis is to build a dynamic model that simulates CA behavior from the structures outlined by the filters and their kinematic properties. At the earliest time-step at which the domain borders are clearly visible in a CA lattice, a dynamic model measures the velocities of the domain borders and capture the outcome of interactions among domains. Next, the model simulates the evolution of the domain borders for the subsequent steps using the measured domain velocities. The model does not use the CA lattice updates—the model simulates only the velocities and interactions properties of the domain boundaries (also referred to as an information-theoretic model). If the behavior of the dynamic model correctly predicts the computational performance of the CA on a large number of randomly initialized configurations, then the evidence supports the hypothesis that the structures highlighted by the filters are the information carrying structures in the lattice.

First, this Chapter will extend Hordijk et al.'s model of information processing from 1DCA (see Chapter 6 for more details) into two dimensions. The results of the filtering methods will be reviewed as the basis for the dynamic model in two dimensions. Level Set and Narrow Band Level Set methods are popular tools for modeling the evolution of two-dimensional interfaces. The hypothesis to be tested here is that Level Set Theory is a good framework for developing a model of particles (contours) in two-dimensions. Section 8.2.1 explores the arguments for why I chose to explore this hypothesis. The second half of this Chapter will introduce these methods, define them in terms of a dynamic model for 2DCA, and outline the issues with their implementation. Finally, Sections 8.3.3 and 8.3.2 show why the hypothesis was determined to be false. I show why it is impossible to predict CA behavior from space-time diagrams using Level Sets or Narrow Band Level Set methods.

8.1 MODEL OF INFORMATION PROCESSING IN 2DCA

The overall approach to characterize the mechanism of computation in 2DCA is the same as Hordijk et al.'s analysis of 1DCA behavior (see Chapter 6 for more details). Such a 2DCA model would simulate the time evolution of the information carrying structures that were identified by the filters. Such an effort requires describing the velocity and deformation of the structures, defining the mechanism of domain interactions, and characterizing the outcome of domain collisions. Figure 8.1 illustrates the necessary steps to build a model of computation for 2DCA. The first two steps demonstrate the application of filtering methods, while the rest of the illustration describes the details of a dynamic model.

Chapter 3 analyzed the behavior of GA evolved rules for 2DCT. The behavior of the rules found by myself, Marques-Pita, and Wolz & de Oliveira shows the lattice settling into domains with well-defined structures (see Appendix A for bit-string representations of these rules). The domains change their shape and location in

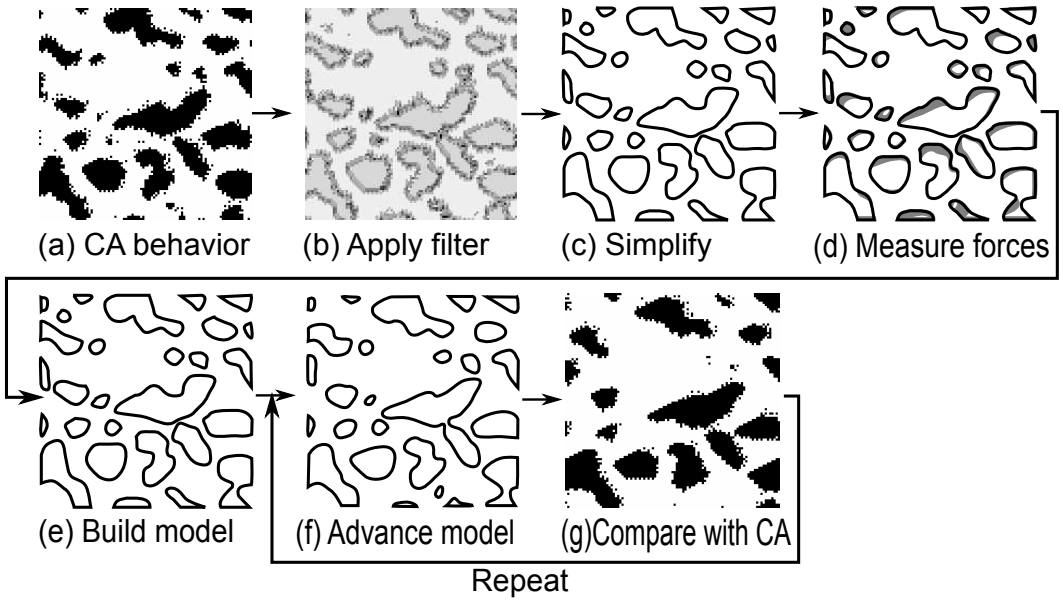


Figure 8.1: Illustration of steps required to build a model of information processing in 2DCA. From left to right: (a) original space-time diagram after lattice settles into black and white domains, (b) the lattice configuration is analyzed by filters to highlight information-carrying structures, (c) domain borders are simplified as single-cell wide lines, (d) the velocity of the domain borders is measured from two space-time diagrams δt time steps apart (showed as gray areas), (e) initial border location and border forces are used to build a model, (f) the domain borders are iteratively advanced using measured forces, and (g) these iterated borders are compared to the borders found when the CA lattice uses the LUT to update its configuration.

space and time. A domain dynamic might result in its annihilation or interaction with another domain (or itself). The domain interactions can be described as fusion, absorption, diffusion, and permeation. The first step in Figure 8.1 shows the CA lattice at the time when the domains are clearly identifiable and their motion is apparent as well—also referred to as the time of measurement. The time of measurement is established empirically, and might come later in time than the condensation time. The *condensation time* is the first time-step when the lattice can be described by the coherent spatio-temporal patterns and border regions in between adjacent patterns (or domains and particles). The difference between the condensation time and time of measurement is the ability to measure the domain's kinematic properties, which might not be visible at the condensation time.

The second step in the process is to identify the coherent spatio-temporal structures in the lattice. The statistically based filters are one way to highlight these structures, since the regular language based filters were not extended to analyze two-dimensional domain patterns. The hybrid filtering method described in Chapter 7 most accurately identified the sites with high versus low information content.

In the next step, the lattice sites with low-information content (sites that simply store information) are subtracted from the lattice. This reveals the sites that propagate information through the lattice — the particles. The hybrid filter highlighted particles as a narrow band, and the simplification step in Figure 8.1 shows particles being simplified by a single-cell wide contour line. A third-degree spline technique can be used to approximate the contour's location.

So far, the model describes the simplified particles and their location in space and time. The next step is to measure the kinematic properties of the contours, so a model can simulate the contour's evolution over time. The kinematic properties are described as a force field that causes the curve's deformation from the current step to the curve's shape and position in the next time step. The acting forces on the curve's surface are marked as the gray areas in the model's sketch. Although

the model definition is incomplete, the space-time diagrams are no longer needed as the model's input. At this time, the model's dependence on the lookup table updates of the CA lattice is severed. See Section 8.3 for a detailed discussion on how to construct the curve deformation force field.

Finally, the Narrow Band Level Set (also known as the Fast Marching Methods) implementation of the Level Set Theory can be used to simulate the evolution and the interaction dynamics of a two-dimensional interface [113]. This robust mathematical model will calculate the contour's next position in space and time by applying the force field to the curve, causing its deformation. This step completes the model's construction. (See Section 8.2 for more details.)

The model will simulate the particle motion and its interactions by updating the curves' location, and advancing the force field along with the curve. The model's accuracy is assessed by comparing the shape and location of the model's contours with the location of actual domains in the CA lattice. If the behavior of the model coincides with the behavior of a CA lattice, then the model describes a mechanism of computation in 2DCA.

8.1.1 Difference of Analytical Scope in 1DCA and 2DCA

It is important to remark that the approach of Hordijk et al. predicts the behavior of a CA rule only at the coarsest analytical level. Specifically, the model was considered predictive of CA behavior if the number of correctly classified ICs was similar. While this measure indicates whether the model would be a good high-level simplification of the CA rule, it does not consider whether it provides the same expressiveness of a given IC, nor whether the expressiveness is true to the system itself.

The motivation for this thesis is to understand how the 2DCA solves a given task, at the scope of the lattice-wide information-carrying structures. A dynamic model based on Hordijk et al.'s analytical scope would have treated 2DCA as a

"black-box" by comparing the performances of a model with that of a rule. The opposite side of the modeling spectrum would be to replicate the exact behavior of a 2DCA on a cell level, but that is not the intention of this work either. The goal of the dynamic modeling in this thesis is to analyze the system on a scale somewhere in the middle – the scale of the lattice-wide information-carrying structures. In the future, a dynamic model at this analytical scope might allow better engineering of CA-like architectures, application of the proposed analysis to other fields, and design of custom built rules to solve a given problem. The utility of such analysis would be difficult to realize if a model would not simulate the velocities and interaction of the highlighted structures.

8.2 BACKGROUND: LEVEL SET THEORY

Level Set Theory (LST) was first introduced by Osher and Sethian [113] as a simple method used to describe the dynamics between two regions on a two-dimensional lattice. The *interface* (Γ) defines a boundary region between two adjacent regions, which may represent objects in a segmented image [62], a boundary between contracting or expanding gases [33], or various multi-phase compressible and incompressible materials [49]. The purpose of this method is to compute the motion of the interface between two environments. Each point of the interface is described by its dynamic properties, interaction rules, and information about the neighboring regions. Figure 8.2 illustrates the motion of the boundary region (an interface) at selected points.

Level Set Theory has been successfully used on digital video segmentation [62], tracking the location of neural stem cell clusters [59], simulation of combustion principles, growth of crystalloid structures, two-fluid flow mechanics, minimal surface and shape recovery, etching and deposition in the micro-fabrication of semiconductor devices [113], and simulation of viscosity and surface tensions in multiphase interfaces [49]. The rendered 3D composition of interface time steps was

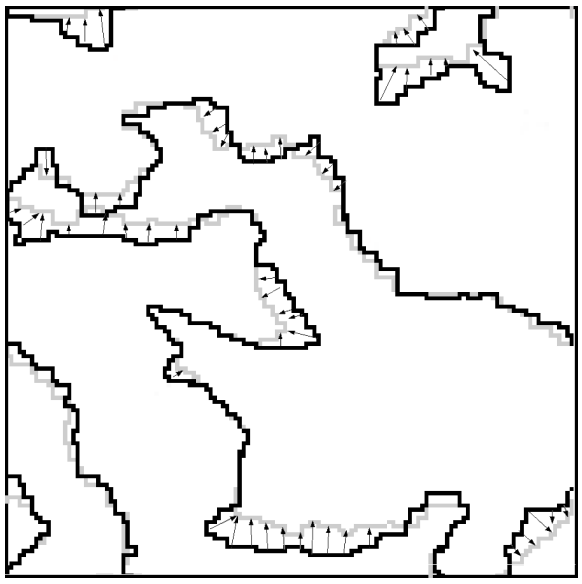


Figure 8.2: Illustration of the level set interface evolution in two dimensions. The interface Γ at time t_0 (black) and at time t_1 (gray). Each point of the interface is assigned both velocity and direction. The arrows in the image display the motion vector for selected points of the interface. The figure shows a GA-evolved CA for the two-dimensional density classification task in two consecutive time steps with the regular domains filtered out manually.

used to create special effects in movies such as Terminator III, Star Wars: Episode III, and Poseidon [33].

8.2.1 Level Set as CA's Dynamic Model

Rather than updating the next position of the contour that represents a particle manually, some of the advantages of using the Level Set Method to calculate position include: the discrete definition of a front, the continuity of the evolving contour, prevention of the swallowtail effect, proper interface collapse, and the ability to propagate acute domain edges. The graphics in Figure 8.3 illustrate these advantages.

The power of Level Set Theory (LST) lies in its flexibility. Each point on the interface is unique in terms of its spatial location and its dynamic properties. LST's

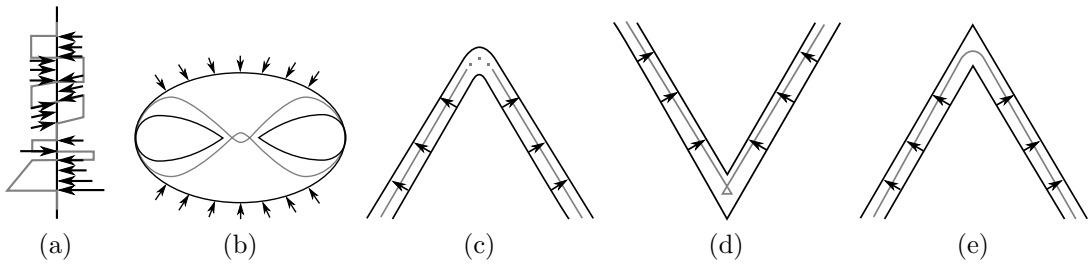


Figure 8.3: Illustration of the advantages of LST shows **a.** discrete definition of a front, **b.** proper interface collapse, **c.** contour continuity, **d.** prevention of a swallowtail effect, and **e.** advancement of a sharp edge. The black arrows mark the forces acting on the contour, and the black lines represent the initial and the final position of a contour. The gray lines show the intermediate contour positions as if the contour would be advanced manually (without the use of LST).

discrete definition of the interface allows the simulation of front advancement with non-uniform velocity. Figure 8.3 (a) illustrates this concept by showing the initial contour as a black line and its deformation using black arrows to mark the forces. The direction and velocity of the evolving front is described by the forces that cause its deformation; the LST does not have any other constraints.

The CA behavior might show domains that shrink with a non-uniform rate of deformation. Before a domain disappears, it might split into smaller sections that will keep decreasing. The LST can simulate such behavior by continuously shrinking an interface until it eventually splits into separate contours. Figure 8.3 (b) shows such behavior.

If the direction of forces is reversed, a contour representing a domain border will expand outward. The expansion of a concave contour is illustrated in Figure 8.3 (c). Updating the position of a contour manually would result in a discontinuity in a place of expansion. The LST guarantees contour continuity as well as proper scaling of a contour's shape by increasing the grid resolution in places of contour expansion (also called adaptive mesh refinement [113]). The same technique is used to accurately propagate other curve features such as an edge of a domain with a sharp edge (e).

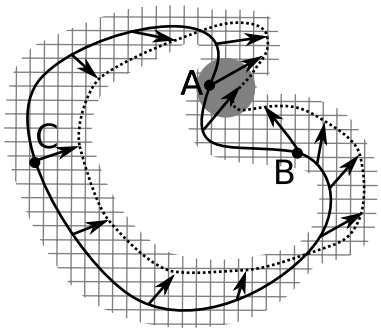


Figure 8.4: Illustration of a Narrow Band Level Set evolving a two-dimensional interface that represents a 2DCA particle. The gray grid lines represent a narrow band of points surrounding the interface that require their values to be recalculated. The solid gray circle marks grid points that the contour sections marked A, B, and C will pass through.

The last example where LST simulation is computationally beneficial is presented by a shrinking domain with two non-parallel borders moving towards each other. The reduction of the level set points that represent such a domain border is needed when two parts of a contour travel towards each other. The swallowtail effect, shown in Figure 8.3 (d), occurs if each point of a level set is updated manually, regardless of the contour's original shape. The LST will automatically reduce the number of points where a curve contracts, which will prevent the "criss-crossing" of the contours and creation of a swallow-tail like shape.

8.2.2 Narrow Band Level Set (NBLS)

The above section illustrated the utility of Level Set Theory to simulate evolution of an interface in two dimensions. It is crucial to notice that the deformation was done by forces originating in each point of the contour. The forces are independent of the local properties of the front (such as its curvature and normal direction), as well as the contour's global properties (such as its relative position on the lattice). This means that the forces cannot be inferred from the curve's shape and location, but have to be determined from the CA dynamic.

Furthermore, the deformation forces can be assigned only to the grid points that represent the contour itself. In other words, the forces cannot be pre-calculated for the entire lattice. If two points of a contour lie on the same trajectory, the grid points would have to be assigned two different force values to account for the velocity of each contour. The solid gray circle in Figure 8.4 marks such a location. The contour sections A, B, and C will all pass through the grid points in the gray circle. Each grid point in the gray circle would have to represent at least three force vectors needed to advance the appropriate contour segment. Since each grid point can hold only a single vector value, the forces have to be propagated through the lattice along with the advancing front. This is the main reason why the forces can be assigned only to a narrow band of grid points around the evolving contour. After the contour position is updated (at the next time step), the force field has to be reinitialized for the new contour position. The *Narrow Band Level Set* (NBLS) refers to the narrow margin of grid points that are subject to an update. Figure 8.4 shows this set as gray grid lines surrounding the contour.

All GA-evolved rules for the 2D density classification task using the $r = 1$ Moore neighborhood have black and white domains with the movement velocity not exceeding three cells per update. The NBLS algorithm has to compute the force field for a three cells-wide band around each domain contour-line and can ignore the rest of the lattice.

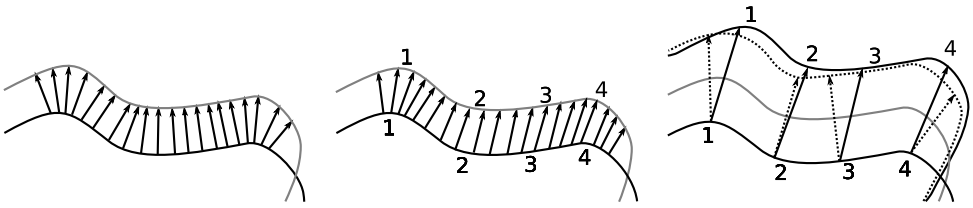
8.3 MEASURING THE INTERFACE VELOCITIES

Before diving into the problem of measuring the force field that models the particle deformation, let's step back and look at the categories of existing LST applications and compare them with the requirements of an information-processing model for 2DCA. The LST was designed to simulate the propagation of a 2D interface in a physical system. The numerous applications include modeling of ocean waves, gas expansion, image segmentation, object disintegration, and flame visualization.

With only a few exceptions, each application falls into one of two categories. The applications in the first group use an explicitly defined 3D object while the level set represents a two-dimensional cut of the object. For example, a gray-scale image segmentation is performed by LST by embedding the two-dimensional image into a three dimensional space. The gray value of each pixel is interpreted as the value in the third dimension. A level set will outline the boundaries of an object in the image by placing a small initial level set inside of an object, then expand the level set interface using a constant polar force field originating inside of the set. The contour will advance until the pixel gradient is too high or the contour's plasticity won't allow its further advancement. The gray pixel value servers as a barrier for the front's advancement, while the motion of the level set "climbs" the explicitly defined 3D surface.

The second category of applications include simulations of well-understood environments. One such example is the breaking of an object due to an impact. After initial force is applied to a 3D model of an object, the LST simulates the energy propagation through the object and updates the bonds between the points that represent given object. The laws of force propagation and material properties of the object are well known and understood. The state and velocity of each object point is updated using the model's energy equations. Most importantly, the environment does not have to be homogeneous, but the behavior of each point of the model is governed by the same set of principles.

The simulation of a 2DCA's dynamic properties seems to be in a category of its own. First, there is no explicit 3D surface to follow. Instead, the velocity of the contour's deformation is measured from the observed CA behavior. The 3D "object" is iteratively constructed by advancing a domain's border using the measured velocities. The 3D "object" in this case represents a domain's shape and interaction with other domains over time. Second, the interface advances in a non-linear fashion. A configuration of each neighborhood is updated by a lookup



(a) Forces normal to tangent

(b) Point correspondence forces

(c) Curve advanced by either force

Figure 8.5: Illustration of measuring forces for an advancing front. The black curve shows a contour's initial position, while gray depicts the contour's location in a later time. **(a)** Forces are measured as the shortest distance between the two contours at each point. **(b)** Forces are measured as the distance between the corresponding features in the starting contour and the feature location in the advanced contours. The features mark 1: convex apex, 2: concave apex, 3: flat section, and 4: convex apex. **(c)** The dotted contour marks the position if normal forces were used to advance the initial contour. The solid curve shows the contour position if the correspondence forces were used; this contour location also marks the actual location of the domain border in the CA lattice.

in a rule table that is 512 bits long. In terms of the lattice dynamics, let's think of the update options as 512 degrees of freedom that are interpreted as the front's motion in two-dimensional space. The following sections will explain the non-linear border dynamics in more detail.

8.3.1 Solving the Correspondence Problem

As described earlier, the forces acting on a contour are independent of local curve properties such as curvature and the direction normal to the contour, as well as the global properties, such as the domain's relative position with respect to the rest of the lattice. The direction of a contour's deformation has to be solved for each point independently.

Figure 8.5 (a) and (b) show two ways of measuring forces that caused a curve's deformation from its initial shape to its subsequent shape at time $t = t + 1$. Although each measuring method can produce different deformation forces, the shape of a curve at time $t = t + 1$ should be identical regardless which set of

forces was used. How accurately the forces predict the shape of an evolving front is assessed by further advancing the curve (multiple time-steps) and comparing the contour's shape with the shape of the actual domain in the CA lattice (Figure 8.5 (c)).

A point-wise measurement of the closest distance between two curves is an obvious way to assign the forces that caused the curve's advancement. At each point, the distance between two curves can be calculated as a normal distance to the curve's tangent or as a Hausdorff distance [91]. Figure 8.5 (a) shows measuring the deformation forces between two curves as a normal distance. Although the LST is designed to maintain the curve's continuity, it will not maintain the curve's shape if the measured forces are sparse, as shown in the figure. The resulting curve will be jagged if the forces are not populated for each point on the destination curve. Due to a contour's expansion, additional force interpolation is required to fill in the concave segments of the curve. For the purpose of a dynamic model, the forces measured as a normal or a Hausdorff distance will result in low accuracy of predicting future contour shapes and positions. The subsequent contour shapes simulated by the LST model will drastically differ from the shape of the original domain border (Figure 8.5 (c)).

The ideal method of measuring the contour's advancement would describe the force on each point as a correspondence between its original location and its new location on the curve at a later time. This type of measurement is referred to as a correspondence force, since it aims to match each point on the original curve with its location on the curve time $t + 1$ (Figure 8.5 (b)). One way to achieve this is to measure the distance between pairs of matching features on the initial and the advanced curve (marked as features 1-4 on Figure 8.5 (b)). The points in-between the features will have forces approximated with respect to the forces measured for the neighboring features. This approach is difficult to implement because: (1) the hybrid filter highlights the domain borders by a narrow band which smooths the

original features of a domain border, (2) the precise location of features, such as an apex of a broad concave contour, is unclear, (3) a domain border has no clear edge; instead a collection of sites is interpreted as a front location (4) a domain border advances in a non-linear fashion, which makes modeling of the front advancement by estimating the deformation forces by linear vectors insufficient, and (5) a domain border can have complex behavior with no apparent features and no velocities, yet forces will originate in this region and propagate into the rest of the border. Since the last two points attest to more than just implementation difficulties, the following sections will discuss these two problems in more detail.

Figure 8.5 (c) shows how the above-described techniques predict the curve's shape and position in a subsequent time-step. The dotted line represents the curve position advanced by the normal forces, while the solid line was rendered using the correspondence forces. In this example, using the correspondence forces yields the curve closet to the shape of the actual domain border.

8.3.2 Noise versus Information

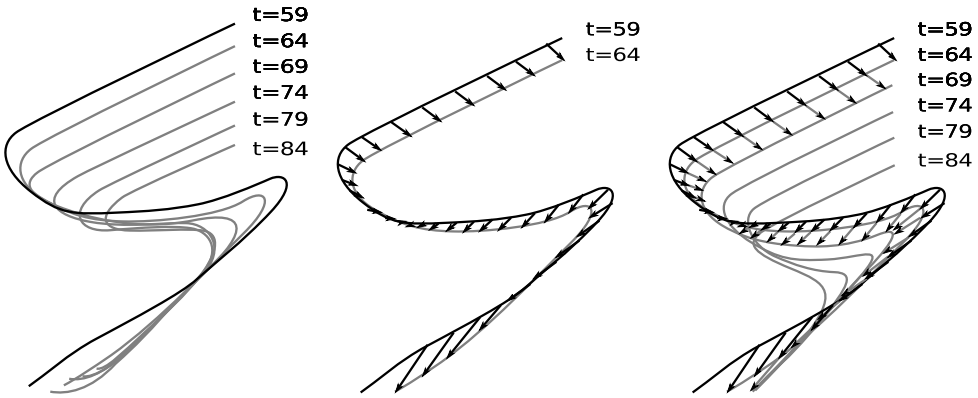
Even though solving the correspondence problem will render accurate initial behavior of the dynamic model, it will not account for the non-linear advancement of the domain border. Figure 8.6 shows an example illustrating an abrupt stop in the velocity of an advancing front and Figure 8.7 compares the contour shapes simulated by a model using the correspondence forces with the shape of a domain in the 2DCA lattice.

Chapter 7 discussed various 2DCA filters and their ability to highlight domain borders. The role of individual sites was not analyzed with respect to the front's motion. In other words, the filters abstracted a collection of sites that formed an advancing front into a domain border. From the perspective of the domain border's dynamics, the original observation that the border motion in the GA evolved rules is linear with noise might have been premature. For example a

(a) $t = 59$ (b) $t = 64$ (c) $t = 69$

(d) $t = 74$ (e) $t = 79$ (f) $t = 84$

Figure 8.6: Cenek's rule for the two-dimensional density classification task. Lattice configurations are shown at times $t = 59, 64, 69, 74, 79$, and 84. A section of a black domain has a border marked with a gray line that suddenly stops advancing. The domain border was outlined manually.



(a) Original border locations

(b) Measure forces

(c) Advanced border by forces

Figure 8.7: A comparison of a domain behavior in a CA lattice (Figure 8.6) with a simulation of the domain border by a model using the correspondence forces. **a.** Border locations in the original CA (Figure 8.6) after the border segments were stacked on top of one another. **b.** Solving the correspondence problem and assigning the deformation forces to the border (black arrows). **c.** Location of the border by advancing its original location using the correspondence forces. The subsequent contours were attained by advancing the forces to their next location (gray arrows). Notice the shape difference between the CA border locations in (a) and the contours simulated by advancing a model at times $t = 74, 79$, and 84 (b)

black site on a white background in the vicinity of a domain border is perceived as noise. These sites are not left-overs from the random initial configuration; instead they are being constantly generated (and consumed) by the updates of sites that make up the noisy domain border. The LC filters will place such a site into a causal state that represents the domain border; the IS and IT filters will also highlight such a site as the domain border due to its low statistical count. Although filters group such sites with the sites of the border region, the CA lattice updates will use this seemingly noisy site for an update in all of its neighboring sites (including its own value). The updates that use this noisy site in a vicinity of the domain border might cause a sudden change in the border's dynamics. After a close examination of the CA behavior in Figure 8.6, it appears that the sites that were originally considered noise around the domain border caused the abrupt stop of the border's advancement. This change was not a gradual attenuation of the front's advancement, but a sudden stop.

Figure 8.7 shows the difference between the behavior of the domain's border in the CA lattice (a) and the behavior of a border contour simulated by a model (c). The model used the correspondence forces (b) to calculate the contour's advancement. The domain border in the CA will unexpectedly stop advancing at step $t = 74$, while the correspondence forces in the model keep advancing the contour in subsequent steps $t = 74, 79$, and 84 (Figure 8.7 c).

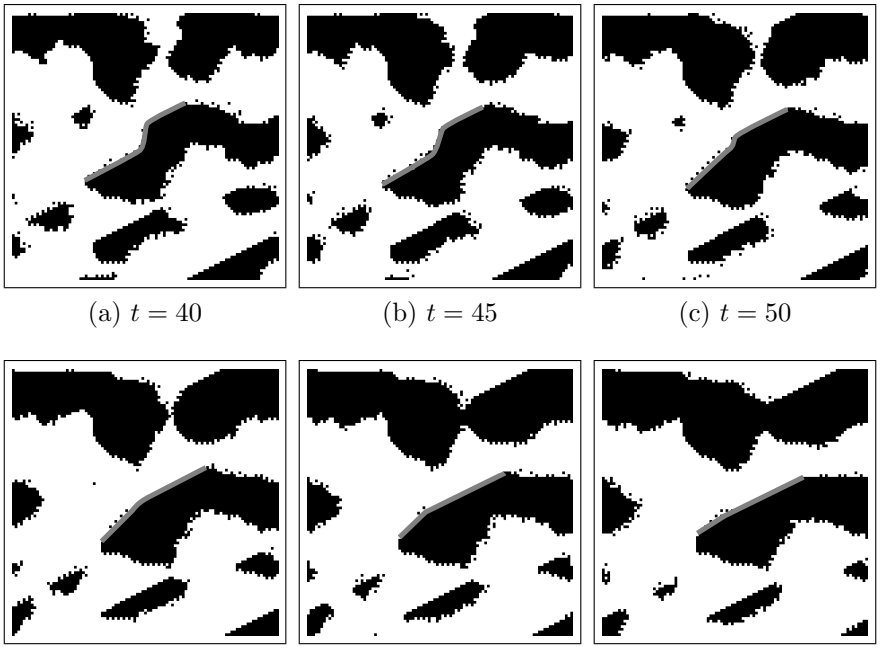
This example illustrates that the abstraction of a domain's border as a smooth contour that propagates information is lossy for the purpose of building a dynamic model. The filters will categorize the site configurations around the domain's front as part of its border, but the same sites were used by CA updates and they caused an unexpected change in the domain's velocity. Since the dynamic model does not use the LUT to predict the front's velocity, it could not predict the sudden stop of the domain's advancement.

8.3.3 Hidden Forces and Complex Regions

The CA behavior during the initial lattice updates is highly complex. The condensation time is the first time-step of the CA configuration at which the lattice behavior can be described by coherently formed domains and particle regions in-between these structures. It is unclear what exactly happens during the initial lattice behavior, but the outcome is clear: an "orderly" CA behavior with well-formed domains that propagate through the lattice. The construction of a model has assumed that the lattice behavior after the condensation time would not slip back into "chaotic" behavior. Figure 8.8 shows the occurrence of highly complex behavior at the domain border which appears much later than the condensation time. The region with complex behavior did not originate from a collision between two domains, but appears in a domain border, between two well defined, linear border segments.

Figure 8.9 (a) shows the CA behavior of an advancing domain border for the same CA illustrated in Figure 8.8. In the movement from time-step $t = 40$ to $t = 45$, the contour features are clearly visible and the correspondence forces agree with the direction of front's movement (shown as the illustration (b)). At time $t = 50$, the behavior of the region inside of the circle changes unexpectedly. The domain border moves in the opposite direction from its original velocity. The left side of the border region changes its pitch and stops moving in the subsequent updates. There are no other domain features or interactions with other particles that caused this change in behavior. This observation supports the hypothesis that the changes in the border's behavior were caused by the forces that originated in the highly dynamic region shown inside of the circle. The previously assessed correspondence forces could not predict such behavior. Figure 8.9 (c) shows the shape of the contours as predicted by a model using correspondence forces.

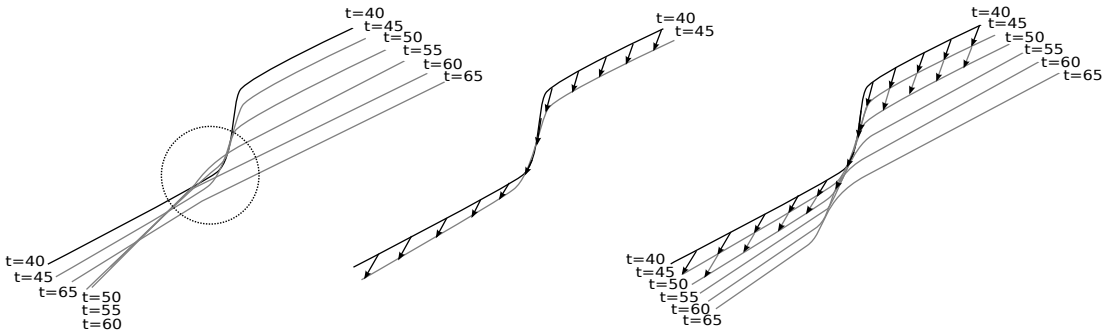
It appears that the border's complex region caused the stagnation of the border's left segment. The size of the complex region continuously decreases between



(a) $t = 40$ (b) $t = 45$ (c) $t = 50$

(d) $t = 55$ (e) $t = 60$ (f) $t = 65$

Figure 8.8: Cenek's rule for the two-dimensional density classification task. Lattice configurations are shown at times $t = 40, 45, 50, 55, 60$, and 65. A section of a black domain has a border marked with a gray line that suddenly reverses its direction of advancement, stops for several steps, and then starts moving again. This border region is located approximately in the middle of the highlighted domain border. The domain border was outlined manually.



(a) Original border locations (b) Measure forces (c) Advanced border by forces

Figure 8.9: A comparison of a domain behavior in the original CA (Figure 8.8) and the simulation of a domain border by a model using the correspondence forces. **a.** Shows the border locations in the original CA (Figure 8.8) after the border segments were stacked on top of one another. The circle in the middle of the outlined border points to a region with complex behavior. The domain border originally moved from left to right, then retracted to its original position, did not advance for a couple of steps, and resumed its motion in steps $t = 60$ and 65. **b.** Shows how to solve the correspondence problem and how to assign acting forces to the border (black arrows). **c.** Shows the location of the border by advancing its original location using the correspondence forces. The subsequent contours are attained by advancing the correspondence forces to their next location (gray arrows).

time $t = 50$ and $t = 65$, because the right side of the domain border keeps advancing. The complex region disappears when the advancing segment on the right catches up with the motionless segment on the left. Just before this happens at time $t = 65$, the left side of the domain border resumes its original pitch and velocity. What caused this change of motion? It appears that when the size of the complex region was the same as the neighborhood diameter (three sites), the moving contour segment sent a "signal" into the motionless contour segment, prompting advancement of this otherwise stagnant part of the contour. This can explain the origin of new forces in the advancing front. More importantly this force or signal was not present in the originally measured force field and it could only originate for the actual output bits stored in the rule's LUT.

8.4 CONCLUSION AND DISCUSSION

Although the Narrow Band Level Set method is a useful tool to simulate evolution of a two-dimensional interface in many problems, it will not accurately predict the behavior of a 2DCA lattice. A dynamic model of information processing would benefit from the NBLS's utilities (such as resolving interactions between particles, shrinking and expanding of a front, etc.), but the forces used by the model to calculate the front's deformation can not be inferred from the velocities of the domain boundaries. Finding a solution to the correspondence problem between two curves would create the most accurate force field to simulate the initial curve deformation, but the forces can not predict the unexpected border behavior.

In this thesis, the goal of the dynamic modeling is to build a computational mechanics framework to describe the mechanism of collective computation in 2DCA. It is possible that a dynamic model has a wrong definition of curve deformation, but still gets the same performance as the CA lattice (as Hordijk et al.'s model for 1DCA). I proposed a stricter definition of the dynamic model; one that requires accurate prediction of the highlighted two-dimensional structures in 2DCA. This

requirement would allow for a more informative understanding of the information-carrying patterns in a lattice.

Information modification in a 1DCA model is expected only where two particles collide. Each particle in one dimension represents a border between two domains, and a particle collision represents processing of information between two or more different domains. Although the collisions between domains in 2DCA have an analogous meaning, the examples in Sections 8.3.2 and 8.3.3 illustrate additional information modification in a domain border itself. The non-trivial border behavior caused a section of the domain border to unexpectedly change velocity and direction. This behavior cannot be predicted by the correspondence forces that were inferred from the CA's space-time behavior after its initial unstructured phase. The unexpected border behavior can occur at much later updates (times $t = 59$ and $t = 40$), and the phenomenon can be explained only from the observations of a CA lattice. Although the counterexamples explain why the model could not predict the curve deformation that represents a domain border, the potential use of NBLS to simulate CA behavior should not be disregarded. There might be a different way to calculate the forces to accurately predict a CA's behavior. Due to a non-linear motion of the domain borders, the behavior of a dynamic model that uses the correspondence forces is different from the behavior of the information-carrying structures highlighted by the filters.

An alternative approach of computing forces for the NBLS-based dynamic model is to use the lookup table itself. In other words, the velocity and the direction of the domain border would be inferred from the CA rule. This approach would have to interpret the meaning of the LUT bits with respect to the border behavior. Such analysis is very difficult because it relies on bridging two analytical scopes. The rule table describes the output bits for a cell's update — a micro scope, while the dynamic model simulates the behavior of the lattice-wide patterns — a macro scope. The analysis would use principles from a micro scope to explain

phenomenon on a macro scope level. This would require solving fundamental issues such as reducing degrees of freedom (from a 512 bit LUT to a two-dimensional vector space), mapping between different domains (bits in a LUT to a force field in a dynamic model), and translating the updates of multiple individual sites to a collective-behavior representative of a border motion. The difficulty of this approach is comparable to connecting the meaning of alleles to explain the cause of genetic disease in biology, the principles of molecular self-assembly to explain the surface properties in material science, and the laws that govern quantum mechanics to explain the macroscopic system behavior in physics.

Chapter 9

RELATED WORK

In general, it is unknown how to effectively "program" CAs to perform computations or what are the best information-processing dynamics in CAs that would accomplish a task. This chapter gives a brief summary of related work on different approaches for finding CA rules for given tasks as well as methods for analyzing rule performance. Previous work on early models of computation in CA can be found in Section 2.2 while Section 4.2 gives background on different strategies to evolve CA rules with GA. The following sections summarize Andre et al.'s genetic programming approach to evolve CA rules, Sipper's parallel cellular machines as an alternative definition of CA and his approach to program these systems, resource sharing technique as an alternative way to preserve genetic diversity in evolving populations, and Marques-Pita's re-description of a binary rule representation. The final section contains brief mention of other related research.

9.1 GENETIC PROGRAMMING

Andre et al. [2] applied genetic programming (GP), a variation of GAs, to the density classification task. GP methodology also uses a population of evolving candidate solutions, and the principles of reproduction and survival are the same for both GP and GAs. The main difference between these two methods is the encoding of individuals in the population. Unlike the binary strings used in GAs, individuals in a GP population have tree structures, made up of *function* and

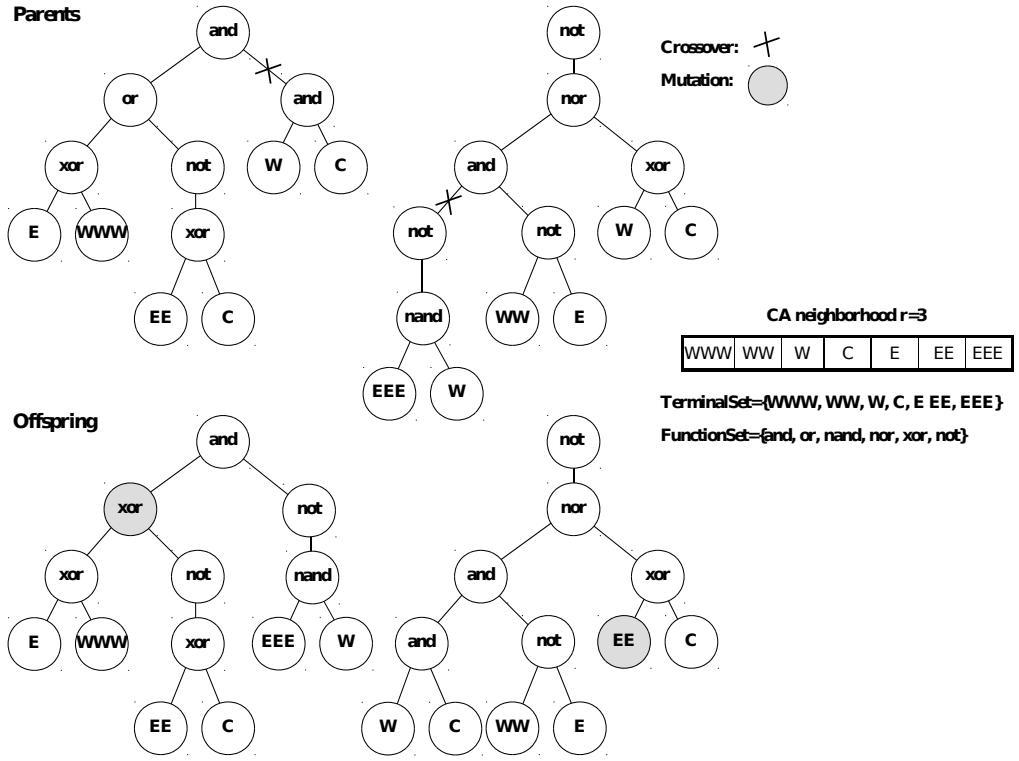


Figure 9.1: An example of the encoding of individuals in a GP population, similar to the one used by Andre et al. [2]. The function set here consists of the logical operators {**and**, **or**, **not**, **nand**, **nor**, and **xor**}. The terminal set represents the states of cells in a 1DCA neighborhood, here {**C**enter, **E**ast, **W**est, **E**astOf**E**ast **W**estOf**W**est, **E**astOf**E**astOf**E**ast, **W**estOf**W**estOf**W**est.} The figure shows the reproduction of $Parent_1$ and $Parent_2$ by crossover with subsequent mutation to produce $Child_1$ and $Child_2$. Reprinted from [15].

terminal nodes. The *function* nodes (internal nodes) are operators from a predefined function set, and the *terminal* nodes (leaves) represent operands from a terminal set. The fitness value is obtained by evaluating the tree on a set of test initial configurations. The crossover operator is applied to two parents by swapping randomly selected sub-trees, and the mutation operation is performed on a single node by creating a new node or by changing its value (Figure 9.1) [64, 65].

The GP algorithm evolved CAs whose performance is comparable to the performance of the best CAs evolved by a traditional GA.

Unlike traditional 1DCAs that use crossover and mutation to evolve fixed length

genome solutions, GP trees evolve to different sizes or shapes, and the subtrees can be substituted out and added to the function set as automatically defined functions. According to Andre et al., this allows GP to better explore the "regularities, symmetries, homogeneities, and modularities of the problem domain" [2]. The best-evolved CAs by GP revealed more complex particles and particle interactions than the CAs found by the EvCA group [24, 51]. It is unclear whether the improved results were due to the GP representation or to the increased population sizes and computation time used by Andre et al.

9.2 PARALLEL CELLULAR MACHINES

The field of evolving CAs has grown in several directions. One important area is evolving non-homogeneous cellular automata [47, 119, 120, 130]. Each cell of a non-homogeneous CA contains two independently evolving chromosomes. One represents the LUT for the cell (different cells can have different LUTs), and the second represents the neighborhood connections for the cell. Both the LUT and the cell's connectivity can be evolved at the same time. Since a task is performed by a collection of cells with different LUTs, there is no single best performing individual; the fitness is a measure of the collective behavior of the cells' LUTs and their neighborhood assignments [118, 120].

One of many tasks studied by Sipper was the global ordering task [119]. Here, the CA has fixed rather than periodic boundaries, so the "left" and "right" parts of the CA lattice are defined. The *ordering* in any given IC pattern will place all 0s on the left, followed by all 1s on the right. The initial density of the IC has to be preserved in the final configuration. Sipper designed a *cellular programming algorithm* to co-evolve multiple LUTs and their neighborhood topologies. Cellular programming carries out the same steps as the conventional GA (initialization, evaluation, reproduction, replacement), but each cell reproduces only with its local neighbors. The LUTs and connectivity chromosomes from the locally connected

sites are the only potential parents for the reproduction and replacement of cell's LUTs and the connectivity tables respectively. The cell's limited connectivity results in genetically diverse populations. If a current population has a cell with a high-fitness LUT, its LUT will not be directly inherited by a given cell unless they are connected. The connectivity chromosome causes spatial isolation that allows evolution to explore multiple CA rules as a part of a collective solution [119, 120].

Sipper exhaustively tested all homogeneous CAs with $r = 1$ on the ordering task, and found that the best performing rule (rule 232) correctly ordered 71% of 1000 randomly generated ICs. The cellular programming algorithm evolved a non-homogeneous CA that outperformed the best homogeneous CA. The evolutionary search identified multiple rules that the non-homogeneous CA used as the components in the final solution. The rules composing the collective CA solution were classified as state preserving or repairing the incorrect ordering of the neighborhood bits. The untested hypothesis is that the cellular programming algorithm can discover multiple important rules (partial traits) that compose more complex collective behavior.

9.3 RESOURCE SHARING

The spatial extension of evolution and coevolution algorithms, discussed in Sections 4.2.2 and 4.3, is used to maintain higher genetic diversity during evolutionary search. Resource sharing is another method for preserving diversity in evolving populations. This method can use a single population view, or be applied on more than one coevolving populations. The algorithm views training examples as a "resource" that is shared among the candidate solutions being evolved. Resource sharing can be applied on most of the algorithms described in this chapter [40, 58, 106, 133, 135].

The resource sharing method defines a host's fitness based on the number of

successful evaluations of test cases and on how many other hosts successfully evaluated these tests. $Fitness(h) = \sum_{j=Tests} \frac{1}{N_j}$ where *Tests* are the successfully evaluated test cases by host h, and N_j is the number of other hosts that correctly evaluated the tests j. Let's look at an example: a host successfully evaluated three ICs out of five. The three test cases were successfully evaluated by five, six, and two other hosts respectively. The fitness value for this host will be the sum of one-fifth, one-sixth and one-half (Figure 9.2). The intuition behind resource sharing is that the hosts that defeat a rarely defeated test case will be awarded by large fitness fraction. This makes the hosts more viable to reproduce and their scarce genetic trait will have a better chance of being passed on to offspring generation.

Resource sharing was one of the adaptive fitness strategies used by Oliveira et al. that found the best-performing rules on density classification and global synchronization tasks [94]. Their approach also yield large quantity of high performing rules for these tasks, which supports the theory that the resource sharing helps preserve genetic diversity in evolving populations.

9.4 AITANA

In Section 5.4 we briefly stated a hypothesis that there might be a connection between the rules' structure and symmetry and the rules' behavior and performance. This observation was based on the ability to reverse some rules that solve the density classification task and end up with rules that solve the global synchronization task.

Marques-Pita et al. explored the notion of rule symmetry and was able to redescribe a binary array representation of a given rule by a compact set of schemas using the rule's structure and symmetry [79, 81, 82, 83]. A schema basically refers to a lookup table entry. It has a size of a neighborhood diameter including the cell that is being updated, and in addition to the binary states $0, 1$ it also uses a wild-card character. The wild-card character # allows for a single schema to describe

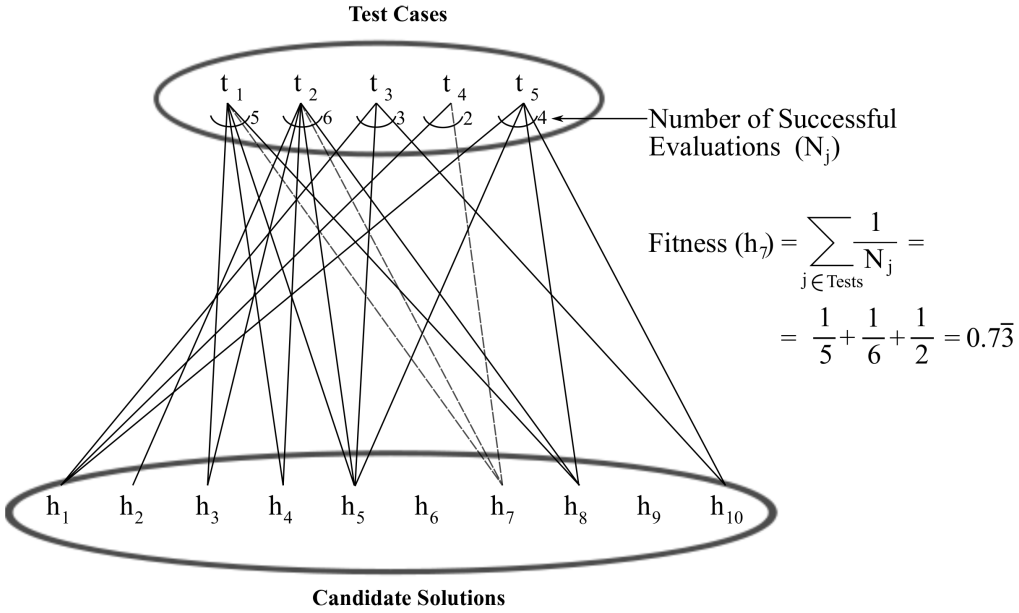


Figure 9.2: An example of a resource sharing fitness evaluation. The edges between the test cases and the candidate solutions denote successful evaluations of tests by candidate solutions.

multiple update patterns where a particular bit value does not influence the update output. The schemas were further subdivided into two sets: a generation set has schemas that always update the current site to 1 and an annihilation set has schemas that always update the current site to 0. An example of a generative schema $\{1, 0, 1, 0, \#, \#, \#\}$ for $r = 3$ 1DCA will update the center site to 1 for any neighborhood where the state of the center site is 0 and the left neighborhood has values $1, 0, 1$. The configuration of states on the right of the current site does not matter.

Aitana is the name of the program that discovers such schemas. It not only provides a compact representation of a binary rule table with high symmetry and structure, it also augments the search space for potentially more efficient search. Most importantly, it introduces a notion of conceptual structure in a rule that might reveal a connection between the rule's behavior and the bits in the lookup table.

9.5 OTHER RELATED WORK

All of the related work so far presented CA that were designed to classify particular lattice configurations. A rule table represents a "program" or a "set of instructions" while the initial lattice configuration is an input instance to be classified. An alternative CA definition uses both the lattice configuration and rule(s) to encode the automata [20, 39].

Ripps used the maximum economy of means to evolve rules for 2DCA [105]. He co-evolved the initial lattice configuration and an LUT to find CA with as many periodic regenerating structures on the lattice as possible. The main contribution of this research is in its novel fitness function used by a GA. An individual's fitness is increased every time the individual reuses rules that already exist in a rule-set. Fitness is decreased for an individual that has to add new rules.

The same CA definition as above was used by Sapin et al. in their approach to automatically discover lattice structures that simulate the operation of an AND gate [109, 111]. Genetic algorithms and Tabu search were used to find gliders, guns, and spatial configurations of these two structures. The local interactions among the discovered structures represent the operation of an AND gate. The contribution of this approach is in its progressive refinement of partial results that evolves the overall lattice configuration towards a desired solution. The broader impact of this research might lead collision-based computing towards automatic design of a universal cellular automaton [20, 110, 111].

Chapter 10

CONCLUSION

10.1 CONTRIBUTIONS

I. I presented evidence that CA are capable of solving proposed problems by emergent global behavior

In this dissertation I propose several novel tasks that challenge the ability of 2DCA to solve a problem by emergent system behavior. Genetic algorithms evolved high performing rules that solve these tasks. The rules had complex behavior that was different for each task. The lattice behavior for the two-dimensional density classification and global synchronization tasks had a global character with information-carrying structures propagating through the entire lattice. The remaining tasks had much smaller features which resulted in solutions with information carrying structures propagating over relatively short distances with shorter lifespan of actively moving structures than the solutions for the classification tasks.

Although GAs were able to find high-performing rules that solve the proposed tasks, I cannot conclusively state to what extent CA are capable of emergent global computation. Without a theoretical framework, proving CA's computational capability can be argued only by using experimental methods. The ability of CA to solve problems must be shown for many more, fundamentally diverse, and globally challenging tasks.

II. I extended statistically based filters to detect coherent spatio-temporal structures in 2DCA space-time behavior.

I extended the analysis of information processing from 1DCA into two-dimensions.

The first step towards understanding the mechanisms of computation in 2DCA was to identify the spatio-temporal structures in the lattice that were informationally relevant. The investigation of the statistical based filters proposed by Shalizi et al. and Lizier et al. led to the definition of filtering techniques for two-dimensional lattices. The implementation of the filters did not produce high quality results. The filters highlighted the information carrying structures with wide blurry borders that resulted in loss of detail, failure to accurately of outlining 2D structures, and no success in highlighting the sections of the domain borders with zero-velocity. I proposed, implemented and tested a hybrid filtering approach that combines these two filtering techniques, which yielded superior results in terms of the accuracy to outline certain 2D structures and the requirements for the computational resources. The hybrid filters identified the domain borders as potential information carrying structures and highlighted them with a narrow border. The structures were highlighted with a continuous border even at the places where the domain border had zero velocity.

III. I showed that construction of a dynamic model of highlighted structures' motion is infeasible from the velocities of the domain boundaries.

Although the filters highlighted the coherent structures in 2DCA, it is only a hypothesis that these structures explain the mechanism of computation in 2DCA. To confirm the meaning of these structures, in Chapter 8 I attempted to build a model of the structures' dynamic behavior. The Narrow Band Level Set methods demonstrated useful properties as a framework for simulating the system's dynamics, but its implementation failed to accurately predict lattice behavior. The model's construction incorrectly assumed that the velocity of the information carrying structures can be inferred from the CA behavior shortly after the lattice condensation time and that this velocity can accurately predict the subsequent

shape and position of these structures in time. The analysis of two counterexamples showed that even if the initial simulation of the domain's motion is accurate (by solving the correspondence problem), the model will not accurately predict the shape of the domain's border. This is because sections of a domain border can depart from their predictable, linear progression through space, and enter into highly complex, unexpected behavior. Such behavior can not be predicted by a model that has its dynamics derived from the lattice behavior (as originally thought).

At this point, the hypothesis that the domain borders identified by the 2DCA filters as the information-carrying structures capture the mechanism of emergent computation in 2DCA can not be confirmed nor denied. A detailed investigation of the domain border behavior revealed that these regions not only propagate (or carry) information through the lattice, but they also process information. The additional information modification in the domain regions occurs even when they do not interact with other domain borders. Due to the information modification by the domain border, the behavior of these structures cannot be predicted by simply measuring their velocities, as was possible in one dimension. An alternative approach is needed to predict the advancement of a domain front in two dimensions.

10.2 EVALUATION OF SUCCESS

Along with the list of contributions, let's briefly examine if the research results fulfilled the proposed research goals.

The CA's computational capability of 2DCA was successfully tested on four benchmark tasks. The GA discovered high-performing rules for classification tasks with performance comparable to the previously published rules. Interestingly, the rules' behavior was unlike previously published rules. This attests to the CA's ability to solve problems in "more than one way". I also evolved CA rules with very different global behavior that perform the image processing tasks. The rule

behaviors that perform these tasks has not been previously reported. Successfully evolving and analyzing rules that solve proposed problems contributes to the hypothesis that 2DCA are computationally capable architectures. The analysis of information processing structures by statistical filtering was previously applied only to 1DCA. It was unknown if this approach would be applicable to 2DCA, what quality of results would it yield, and what would be the computational cost of these filters. I extended the filtering approaches to two-dimensional lattice, and implemented a hybrid filtering approach that outlined information-carrying structures in 2DCA with the highest accuracy, least noise, and acceptable computational cost.

Although I failed to build a model of information processing in 2DCA (the dynamic model), I attribute this failure to unpredictable rule behavior rather than the proposed methodology. I detected two counterexamples that clearly identify reasons why information-carrying structures in 2DCA cannot be modeled from the rule's space-time behavior. Although this contribution does not further explain the mechanism of computation in 2DCA, it points out the non-linear lattice dynamics that cannot be predicted by a model which uses lattice behavior to initialize its dynamics.

10.3 FUTURE WORK AND OPEN QUESTIONS

The filtering results do not provide a thorough account of how CA perform collective information processing; it is unknown how to modify the behavior of the information-carrying structures to design new rules for solving new problems by collective system behavior. There is a missing link among how these structures are formed, what causes their motion, and the encoding of the CA look-up tables. This link is partially characterized by the catalogues of domains and particle interactions obtained with the Computational Mechanics framework, and although this thesis discussed the basics of the Computational Mechanics framework, that

framework currently suffers from drawbacks that impede its applicability to CAs in general. Moreover, no solid connection has been established between the look-up table bits and formation of the information-carrying structures with desired lattice dynamics. Therefore, a current challenge is to derive, from filtered diagrams, the building blocks responsible for computation in the dynamics of CA. These building blocks should be expressed as mathematical formalisms that capture the essential features of local interconnected neighborhoods that *control* the state transitions of cells in a CA lattice, where the dynamic coupling of these building blocks can then be used to explain collective computation, and which can be created or modified to control collective dynamics. Such characterizations of the building blocks of computation in CAs could be used, for instance, to design models of collective computation in nature, such as the collective control of stomata apertures on a plant's leaf (see [100]).

10.3.1 Rule mechanics

The above described approach to reveal the mechanism of collective computation in a system is inferred from the space-time dynamics of a lattice. The study of the CA behavior is empirical in its nature, and can not be generalized as *the* mechanism of computation in 2DCA. This notion is further supported by the analysis of the GA-evolved CA rules for different tasks proposed in Chapter 3. The CAs have very different behavior that articulate widely different mechanisms of information processing in the lattice. Even the rules evolved for the two-dimensional density classification task by Cenek, Marques-Pita, and Wolz & de Oliveira have different behavior from one another, and any conclusions about the nature of the information processing in the system can not necessarily be generalized from one rule to another. As long as the approach to analyze the mechanism of collective computation in the lattice is based on the empirical study of CA behavior, the

conclusions reached are rule specific. A general Computational Mechanics framework is needed that would analyze the structure of the CA rules with respect to the lattice dynamics.

Marques-Pita observed that the bits in the GA-evolved rules for the density classification task form patterns that are repeated throughout the rule table [79, 81, 82, 83]. His symmetry-based rule re-description method uses only the rule table to analyze the CA behavior. Marques-Pita's preliminary work proved to be useful as a rule-based filtering approach to identify the coherent spatiotemporal patterns in a CA lattice. The meaning of a rule's symmetry has to be further investigated in connection to the lattice behavior. An explanation of the structure and pattern of a rule's bits and the meaning of the bits' periodic pattern might identify the behavioral building blocks of a rule. A drastic reduction of a rule's search-space would be achieved by describing the building blocks in terms of their computational function in the lattice, understanding blocks' placement within the rule table, and exploiting the periodic structure of a rule. Constraining the search-space would aid in designing rule to achieve desired behavior, a fast reconfiguration of a CA lattice to correct or change its behavior on-the-fly, and achieving a correct solution for the problems that require multiple rules to solve a problem. In addition to being able to "program" a CA lattice, the analysis of rule performance would not have to rely on the rule's space-time behavior. A rule can be analyzed directly from the structure of the lookup table.

10.3.2 Towards real-life applications

As a partial motivation for this work, several chapters suggested that the future generation of devices will consist of inherently parallel, potentially faulty, locally connected, and decentralized components. The original definition of CA is an idealized mathematical abstraction to study complex system behavior. As a first step towards real-life applications of CA-like devices, the original CA definition has

to be relaxed. Alternative definitions are needed to account for the defects that occur during engineering and assembly, the lack of component synchronization, and the system configuration for a given implementation environment. Such alternative CA definitions can be simulated by an error-prone lattice, non-local component connectivity, and asynchronous update schemas.

The error-prone lattice can be defined by a subset of cells in the lattice that does not update, updates incorrectly, or contains faulty connections to the neighboring sites. Manufacturing systems components that never fail and distributing them on a perfect two-dimensional grid is not realistic, therefore these experiments are important for the future manufacturing and configuration of such systems. Special attention should be paid to how these systems generate faulty information signals, how these signals propagate though the lattice, and if the solutions are robust enough to repair the errors automatically.

The non-local connectivity in the lattice can be simulated by a set of cells having a limited number of neighbors wired to distant sites. It is unknown under what conditions these non-local connections can improve or hinder the ability of the system to perform a given task. Rules evolved using non-local connections will likely show different information processing characteristics than rules evolved on regular lattices. A small-world network is a theoretical model where the system components have non-local and irregular connectivity with other network components [63, 93]. Tomassini et al. evolved a small-world network to perform the one-dimensional density classification and the global synchronization tasks. Their results show that the networks with evolved component connectivity have higher performance than the networks with the system components connected in a regular pattern [127, 128]. Additional research shows that the small-world networks are more robust to the random perturbations of the component inter-connect [26]. A random boolean network (RBN) is even more relaxed model of a discrete dynamical network where the system components are connected at random and a node's

state is updated by a randomly generated logic function [29, 61]. Similar to the small-world network, a RBN was also shown to be capable of solving global tasks that require collective computation in a lattice, and that this type of network is also robust against damage spreading in the system [75, 85].

Finally, the investigation of the asynchronous update schemes would include updating cells in random order, applying updates to a selected subset of sites, and using genetic algorithms to evolve timing schemas to combine more than one rule in a non-homogeneous CA. Using these alternative update configurations might yield systems with more complex behavior or result in superior performance of the architecture versus the individual rules. Several research groups showed that the asynchronous CA are capable of solving tasks that require global cooperation among system components [92, 121, 124, 129].

The study of these alternative system definitions must address the following questions: how computationally capable are these systems, how to control and "program" these systems, and how can their behavior be analyzed? The answers to these questions will likely create applications that use collective system behavior to solve real-life problems as well as allow the behavior analysis of existing networks.

10.4 IN A BROADER CONTEXT...

Just imagine being able to take a large number of sensors, nano-scale devices, or semi-autonomous robots, distribute them in space, and let them evolve or "program" them to solve problems. Applications of such architectures has great potential, such as: massive sensor networks to discover earthquakes, tsunami, and geodetic events; expert architectures to detect features in multidimensional spaces such as carbon structures in alloys or localizing abnormal biological tissue; and nano-scale bots to assemble structures with desired shape and topology such as drug transport agents or device interconnect for future generation electronics. Due

to the massive number of components, asynchronous and parallel nature, nano-scale of the target, or unreliable connectivity and components, these systems can not "compute" desired answers using the traditional von Neumann model of computation. Instead, the system components have to solve the problem collectively by forming complex behaviors.

The research presented in this work was on the ability of cellular automata to perform an emergent collective behavior to compute a task. Let's shift the application domain away from 2DCA. Social organizations, hybrid sensor systems, leaf stomata structure, and CPU architectures are examples of networks that can be viewed as complex systems — locally connected, potentially faulty, decentralized networks of simple components with complex interaction dynamics. Their origin and function might be spontaneous without a job, a task, or an action to perform (such as social networks) or constructed by design to perform a specific task (such as CPUs). The emergent system behavior should be viewed as an appearance of global interaction patterns. Although correct operation of such systems might not depend on formation of these patterns, instead detecting global interaction patterns might reveal useful properties or side-effects of a system (such as congestion, collisions, etc.). Discovering the system-wide patterns, modeling the dynamic properties of these structures, and understanding their role in a system's behavior will undoubtedly lead to better design, control, and use of natural and artificial systems.

REFERENCES

[1] A. Adamatzky, A. Wuensche, and B. De Lacy Costello. Glider-based computing in reaction-diffusion hexagonal cellular automata. *Chaos, Solitons, and Fractals*, 27(2):287–295, 2006.

[2] D. Andre, F. H. Bennett III, and J. R. Koza. Evolution of intricate long-distance communication signals in cellular automata using genetic programming. In *Artificial Life: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA, 1996. MIT Press.

[3] D. Ashlock. *Evolutionary Computation for Modeling and Optimization.* Springer, 2006.

[4] T. Back. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, NY, 1996.

[5] T. Bäck and R. Breukelaar. Using genetic algorithms to evolve behavior in cellular automata. In C. S. Calude, M. J. Dinneen, Paun G., Perez-Jimenez M. J., and Rozenberg G., editors, *Unconventional Computation: 4th International Conference*, volume 3699, pages 1–10, 2005.

[6] D. Basanta, P. J. Bentley, M. A. Miodownik, and E. A. Holm. Evolving cellular automata to grow microstructures. In *Genetic Programming: 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003. Proceedings*, pages 77–130. Springer Berlin / Heidelberg, 2004.

[7] A. Basu, S. C. Lin, C. Wasshuber, A. M. Ionescu, and K. Banerjee. A comprehensive analytical capacitance model of a two dimensional nanodot array. *5th International Symposium on Quality Electronic Design*, 0:259–264, 2004.

[8] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *What is Life?*, chapter 25. Academic Press, London, 1982.

[9] N. Boccara, J. Nasser, and M. Roger. Particle-like structures and their interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules. *Physical Review A*, 44:866–875, 1991.

[10] A. Bucci and J. B. Pollack. Order-theoretic analysis of coevolution problems: Coevolutionary statics. In *GECCO 2002 Workshop on Understanding Coevolution: Theory and Analysis of Coevolutionary Algorithms*, volume 1, pages 229–235, 2002.

[11] A. W. Burks. *Essays on Cellular Automata*. University of Illinois Press, Urban, IL, 1970.

[12] J. Carmonam, J. Cortadella, Y. Takada, and F. Peper. Formal methods for the analysis and synthesis of nanometer-scale cellular arrays. *ACM Journal on Emerging Technologies in Computing Systems*, 4(2):8:1–8:27, 2008.

[13] J. Cartlidge and S. Bullock. Combating coevolutionary disengagement by reducing parasite virulence. *Evolutionary Computation*, 12 (2):193–222, 2004.

[14] M. Cenek. Supplementary data. Web 10-05, Department of Computer Science. Portland State Univerisity., http://www.cs.pdx.edu/research/technicalreports/, April 2010.

[15] M Cenek and M Mitchell. Evolving cellular automata. In R. A. Meyers,

editor, *Encyclopedia of Complexity and Systems Science*, pages 3233–3242. Springer New York, 2009. 10.1007/978-0-387-30440-3_191.

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

[17] P. Chopra and A. Bender. Evolved cellular automata for protein secondary structure prediction imitate the determinants for folding observed in nature. *Silico Biology 7*, 0007:87–93, 2006.

[18] P. S. Churchland and T. J. Sejnowski. *The Computational Brain*. MIT Press, Cambridge, MA, USA, 1994.

[19] E. F. Codd. *Cellular Automata*. ACM Monograph series. Academic Press, 1968.

[20] J. H. Conway, R. K. Guy, and E. R. Berlekamp, editors. *Winning Ways For Your Mathematical Plays, Volume 2*. Academic Press, New York, 1982.

[21] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15/1:1–40, 2004.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[23] J. P. Crutchfield and J. E. Hanson. Turbulent pattern bases for cellular automata. *Physica D*, 69:279–301, 1993.

[24] J. P. Crutchfield, M. Mitchell, and R Das. Evolutionary design of collective computation in cellular automata. In J. P. Crutchfield and P. K. Schuster, editors, *Evolutionary Dynamics—Exploring the Interplay of Selection, Neutrality, Accident, and Function*, pages 361–411, New York, 2003. Oxford University Press.

[25] J. P. Crutchfield and K. Young. Inferring statistical complexity. *Phys. Rev. Let.*, 63:105, 1989.

[26] Ch. Darabos, M. Giacobini, and Tomassini M. Cellular automata scale-free automata networks are not robust in a collective computational task. In S. El Yacoubi, B. Chopard, and S. Bandini, editors, *Proceedings of 7th International Conference on Cellular Automata, for Research and Industry*, volume 4173 of *Lecture notes in computer science*, pages 512–522, Berlin, 2006. Springer Verlag.

[27] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann.

[28] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature-III*, volume 866, pages 344–353. Springer-Verlag, 1994.

[29] B. Drossel. *Random Boolean Networks*, pages 69–110. Wiley-VCH Verlag GmbH & Co. KGaA, 2009.

[30] K. Eloranta. The dynamics of defect ensembles in one-dimesional cellular automata. *Journal of Statistical Physics*, 76(5/6):1377, 1994.

[31] J. Elson, R. Karp, C. Papadimitriou, and S. Shenker. Global synchronization in sensornets. In Martin Farach-Colton, editor, *LATIN 2004: Theoretical Informatics*, volume 2976 of *Lecture Notes in Computer Science*, pages 2705–2705. Springer Berlin / Heidelberg, 2004.

[32] J. D. Farmer, T. Toffoli, and S. Wolfram. *Cellular Automata: Proceedings of an Interdisciplinary Workshop*. Elsevier Science, Los Alamos, New Mexico, 1984.

[33] R. Fedkiw. http://physbam.stanford.edu/ fedkiw/. Web, January 2008.

[34] D. Feldman. Information theory, excess entropy and statistical complexity: Discovering and quantifying statistical structure, 1998. Unpublished lecture notes.

[35] S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature PPSN VI*, 2000.

[36] H. E. Foundalis. *PHAEACO: A Cognitive Architecture Inspired by Bongard Problems*. PhD thesis, Indiana University, 2006.

[37] P. Funes, E. Sklar, H. Juille, and J. Pollack. Animal-animat coevolution: Using the animal population as fitness function. In R. Pfeiffer, B. Blumberg, J. A. Wilson, and S. Meyer, editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pages 525–533. MIT Press, 1998.

[38] P. Gacs, G. L. Kurdyumov, and L. A. Levin. One dimensional uniform arrays that wash out finite islands. *Problemy Peredachi Informatsi*, 14:92–98, 1978.

[39] M. Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, 223:120–123, 1970.

[40] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International*

Conference on Genetic Algorithms on Genetic algorithms and their application, pages 41–49, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

[41] P. Grassberger. Chaos and diffusion in deterministic cellular automata. *Physica D*, 10(1-2):52–58, 1983.

[42] J. E. Hanson. *Computational Mechanics of Cellular Automata*. PhD thesis, University of California at Berkeley, Berkeley, CA, 1993.

[43] J. E. Hanson and J. P. Crutchfield. The attractor-basin portrait of a cellular automaton. *Journal of Statistical Physics*, 66:1415–1462, 1992.

[44] J. E. Hanson and J. P. Crutchfield. The attractor-basin portrait of a cellular automaton. *Journal of Statistical Physics*, 66:1415, 1992.

[45] J. E. Hanson and J. P. Crutchfield. Computational mechanics of cellular automata: An example. *Physica D*, 103(1-4):169–189, 1997.

[46] S. A. Haque, M. Yamamoto, R. Nakatani, and Y. Endo. Magnetic logic gate for binary computing. *Science and Technology of Advanced Materials*, 5/1-2:79–82, 2004.

[47] H. Hartman and G. Y. Vichniac. Inhomogeneous cellular automata (inca). In E. Bienenstock, F. Fogelman, and G. Weisbuch, editors, *Disordered Systems and Biological Organization*, volume F20, pages 53–57. Springer-Verlag, Berlin, 1986.

[48] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[49] J.-M. Hong, T. Shinar, M. Kang, and R. Fedkiw. On boundary condition capturing for multiphase interfaces. *Journal of Scientific Computations*, 31:99–125, 2007.

[50] W. Hordijk. *Dynamics, Emergent Computation, and Evolution in Cellular Automata*. PhD thesis, Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, 1999.

[51] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Embedded-particle computation in evolved cellular automata. In T. Toffoli, M. Biafore, and J. Leão, editors, *Physics and Computation 1996*, pages 153–158. New England Complex Systems Institute, 1996.

[52] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Mechanisms of emergent computation in cellular automata. In A. E. Eiben, editor, *Proceedings of the Fifth International Conference on Parallel Problem Solving From Nature—PPSN V*, New York, 1998. Springer.

[53] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Mechanisms of emergent computation in cellular automata. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature-V*. Springer-Verlag, 1998.

[54] M. Ikebe and Y. Amemiya. VMoS cellular-automaton circuit for picture processing. In T. Miki, editor, *Brainware: Bio-Inspired Architectures and its Hardware Implementation*, volume 6 of *FLSI Soft Computing*, chapter 6, pages 135–162. World Scientific, 2001.

[55] F. Jiménez-Morales, J. P. Crutchfield, and M. Mitchell. Evolving two-dimensional cellular automata to perform density classification: A report on work in progress. *Parallel Computing*, 27 (5):571–585, 2001.

[56] H. Juillé and J. B. Pollack. Coevolutionary learning: A case study. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 24–26, 1998.

[57] H. Juillé and J. B. Pollack. Coevolving the 'ideal' trainer: Application to the discovery of cellular automata rules. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, San Francisco, CA, 1998. Morgan Kaufmann.

[58] J. B. Juillé, H.and Pollack. Dynamics of co-evolutionary learning. In P. Maes, M. J. Mataric, J. A. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, 1996. MIT Press.

[59] N. N. Kachouie and P.W. Fieguth. A narrow-band level-set method with dynamic velocity for neural stem cell cluster segmentation. In *Image Analysis and Recognition*, Lecture Notes in Computer Science, pages 1006–1013. Springer Berlin / Heidelberg, 2005.

[60] D. Kaplan and L. Glass, editors. *Understanding Nonlinear Dynamics*. New York: Springer-Verlag, New York, 1995.

[61] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[62] P. Kehoe and R.B. Reilly. Applying level set theory to digital video segmentation. In *IEEE International Conference on Image Processing: Proceedings of the 2001 International Workshop on Very Low Bitrate Video Coding*, October 2001.

[63] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *In Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, Portland, OR, 2000. ACM.

[64] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[65] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, 1994.

[66] M. Land and R. K. Belew. No perfect two-state cellular automata for density classification exists. *Phys. Rev. Lett.*, 74(25):5148–5150, June 1995.

[67] C. Langton. Studying artificial life with cellular automata. *Physica D*, 10D:120, 1986.

[68] C. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, 42:12–37, 1990.

[69] K. Lindgren, C. Moore, and M. Nordahl. Complexity of two-dimensional patterns. *Journal of Statistical Physics*, 91(5-6):909–951, 1998.

[70] K. Lindgren and M. G. Nordahl. Universal computation in simple one dimensional cellular automata. *Complex Systems*, 4:299–318, 1990.

[71] J. T. Lizier, M. Prokopenko, and A. Y. Zomaya. Detecting non-trivial computation in complex dynamics. In *Proceedings of European Conference on Artificial Life (ECAL '07)*, 2007.

[72] J. T. Lizier, M. Prokopenko, and A. Y. Zomaya. A framework for the local information dynamics of distributed computation in complex systems. Web. http://arxiv.org/abs/0811.2690, November 2008.

[73] J. T. Lizier, M. Prokopenko, and A. Y. Zomaya. Local information transfer as a spatiotemporal filter for complex systems. *PHYSICAL REVIEW E*, 77:026110, 2008.

[74] J. D. Lohn and J. A. Reggia. Automatic discovery of self-replicating structures in cellular automata. In *IEEE Transactions on Evolutionary Computation*, volume 1 (3), pages 165–178, 1997.

[75] Q. Lu and C. Teuscher. Damage spreading in spatial and small-world random boolean networks, 2009.

[76] Cenek M., M. Mitchell, and M. Marques-Pita. Automatic detection of information-processing structures in two-dimensional cellular automata. *Unpublished Manuscript*, 2010.

[77] B. F. Madore and W. L. Freedman. Computer simulations of the Belousov-Zhabotinsky reaction. *Science*, 222:615–616, 11 November 1983.

[78] N. Margolus. Physics-like models of computation. *Physica D Nonlinear Phenomena*, 10:81–95, 1984.

[79] M. Marques-Pita. *Aitana: A Developmental Cognitive Artifact to Explore the Evolution of Conceptual Representations of Cellular Automata-based Complex Systems.* PhD thesis, School of Informatics, University of Edinburgh, Edinburgh UK, 2006.

[80] M. Marques-Pita, 2010. Personal Communication.

[81] M. Marques-Pita, M. Mitchell, and L. M. Rocha. The role of conceptual structure in designing cellular automata to perform collective computation. In C. S. Calude, J. F. G. Costa, R. Freund, M. Oswald, and G. Rozenberg, editors, *Unconventional Computation: 7th International Conference (UC 2008)*, UC '08, pages 146–163, Vienna, Austria, 2008. Springer-Verlag. Volume 5204 of Lecture Notes in Computer Science.

[82] M. Marques-Pita and L. M. Rocha. Conceptual structure in cellular automata: The density classification task. In S. Bullock, J. Noble, R. A. Watson, and M. A. Bedau, editors, *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, pages 390–398, Cambridge, MA, 2008. MIT Press.

[83] M. Marques-Pita, L. M. Rocha, and H. Pain. Conceptual representations: What do they have to say about the density classification task by cellular automata? In *Proceedings of the European Conference on Complex Systems (ECCS'06)*, pages 180–195, 2006.

[84] B. Martin. A group interpretation of particles generated by one-dimensional cellular automaton 54, wolfram's rule. *International Journal of Modern Physics C*, 11(1):101–123, 2000.

[85] B. Mesot and C. Teuscher. Deducing local rules for solving global tasks with random boolean networks. *Physica D: Nonlinear Phenomena*, 211(1-2):88–106, 2005.

[86] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[87] M. Mitchell. Computation in cellular automata: A selected review. In T. Gramss, S. Bornholdt, M. Gross, M. Mitchell, and T. Pellizzari, editors, *Nonstandard Computation*, pages 95–140. Weinheim: VCH Verlagsgesellschaft, 1998.

[88] M. Mitchell, J. P. Crutchfield, and R. Das. Evolving cellular automata to perform computations: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA '96)*, Moscow, Russia, 1996. Russian Academy of Sciences.

[89] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.

[90] M. Mitchell, M. D. Thomure, and N. L. Williams. The role of space in the success of coevolutionary learning. In Luis Mateus Rocha, Larry S.

Yaeger, Mark A. Bedau, Dario Floreano, Robert L. Goldstone, and Alessandro Vespignani, editors, *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, pages 118–124, Cambridge, MA, 2006. MIT Press.

[91] J. Munkres. *Topology*. Prentice Hall, Upper Saddle River, NJ, 2000.

[92] C. L. Nehaniv. Evolution in asynchronous cellular automata. In *Proceedings of the eighth international conference on Artificial life*, pages 65–73, Cambridge, MA, USA, 2003. MIT Press.

[93] M. Newman, A.-L. Barabasi, and D. J. Watts. *The Structure and Dynamics of Networks: (Princeton Studies in Complexity)*. Princeton University Press, 1 edition, 2006.

[94] G. M. B. Oliveira, L. G. A. Martins, L. B. de Carvalho, and E. Fynn. Some investigations about synchronization and density classification tasks in one-dimensional and two-dimensional cellular automata rule spaces. *Electron. Notes Theor. Comput. Sci.*, 252:121–142, October 2009.

[95] N. H. Packard. Adaptation toward the edge of chaos. In J. A. S. Kelso, A. J. Mandell, and M.F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, chapter Part III., pages 293–301. World Scientific, 1988.

[96] L. Pagie and P. Hogeweg. Evolutionary consequences of coevolving targets. *Evolutionary Computation*, 5 (4):401–418, 1997.

[97] L. Pagie and M. Mitchell. A comparison of evolutionary and coevolutionary search. In R. K. Belew and H. Juillè, editors, *Coevolution: Turning Adaptive Algorithms upon Themselves*, pages 20–25, San Francisco, California, USA, 7 2001.

[98] L. Pagie and M. Mitchell. A comparison of evolutionary and coevolutionary search. *International Journal of Computational Intelligence and Applications*, 2 (1):53–69, 2002.

[99] J. K. Park, K. Steiglitz, and W. P. Thurston. Soliton-like behavior in automata. *Physica D*, 19(3):423–432, 1986.

[100] D. Peak, J. D. West, S. M. Messinger, and K. A. Mott. Evidence for complex, collective dynamics and emergent, distributed computation in plants. In R. Schekman et al., editor, *Proceesings of the National Academy of Sciences of the United States of America*, volume 101. 4., pages 918–922. The National Academy of Sciences, 2004.

[101] M. Pivato. Defect particle kinematics in one-dimensional cellular automata. *Theoretical Computer Scienc*, 377(1-3):205–228, 2007.

[102] A. Popovici and D. Popovici. Optimizing epochal evolutionary search: Population-size independent. In J. Rosenthal et al., editor, *15th International Symposium on Mathematical Theory of Networks and Systems*, August 12-16 2002.

[103] Das R. *The Evolution of Emergent Computation in Cellular Automata*. PhD thesis, Colorado State University, Fort Collings, CO, 1998.

[104] R. Reynaga and E. Amthauer. Two-dimensional cellular automata of radius one for density classification task $\rho = \frac{1}{2}$. *Pattern Recogn. Lett.*, 24(15):2849–2856, 2003.

[105] D. L. Ripps. Using economy of means to evolve transition rules within 2d cellular automata. *Artificial life*, 16 (2)(1064-5462):119–126, 2010.

[106] C. Rosin and R. Belew. Methods for competitive coevolution: Finding opponents worth beating. In L. J. Eshelman, editor, *Proceedings of the Sixth*

International Conference on Genetic Algorithms, pages 373–380, San Francisco, CA, 1995. Morgan Kaufmann.

[107] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, MIT Press, 5 (1):1–29, 1997.

[108] P. L. Rosin. Training cellular automata for image processing. In H. Kalviainen, J. Parkkinen, and A. Kaarna, editors, *Image Analysis. 14th Scandinavian Conference, SCIA 2005*, pages 195–204, Joensuu, Finland, 2005. Springer Berlin. Heidelberg. Volume 3540 of Lecture Notes in Computer Science.

[109] E. Sapin. Gliders and glider guns discovery in cellular automata. In *Game of Life Cellular Automata*, volume Chapter 9, pages 135–173. Springer, 2010.

[110] E. Sapin, O. Bailleux, J. J. Chabrier, and P. Collet. A new universal automata discovered by evolutionary algorithms. In *Genetic and Evolutionary Computation Conference*, volume 3102 of *Lecture Notes in Computer Science*, pages 175–187, 2004.

[111] E. Sapin, L. Bull, and A. Adamatzky. Genetic approaches to search for computing patterns in cellular automata. *IEEE Computational Intelligence Magazine*, 4:20–28, August 2009.

[112] A. Schadschneider. *Cellular Automaton Approach to Pedestrian Dynamics—Theory*, pages 75–86. Berlin: Springer-Verlag, 2001.

[113] J. A. Sethian. *Level Set Method*. Cambridge University Press, Cambridge MA, 1996.

[114] C. R. Shalizi. *Causal Architecture, Complexity, and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Wisconsin, Madison, Madison, WI, 2001.

[115] C. R. Shalizi, R. Haslinger, J. B. Rouquier, K. L. Klinkner, and C. Moore. Automatic filters for the detection of coherent structure in spatiotemporal systems. *Physical Review E*, 73:036104, 2006.

[116] Cosma Rohilla Shalizi. Optimal nonlinear prediction of random fields on networks. In *Discrete Mathematics and Theoretical Computer Science, AB(DMCS):11?30*, pages 11–30, 2003.

[117] S. R. Shenoy and A. Daniel. Intel architecture and silicon cadence: The catalyst for industry innovation. Web: ftp://download.intel.com/software/pdf/IAandSiliconCadence.pdf, January 2008.

[118] M. Sipper. Non-uniform cellular automata: Evolution in rule space and formation of complex structures. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 394–399, Cambridge, MA, 1994. MIT Press.

[119] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer, Berlin, Heidelberg, 1997.

[120] M. Sipper and E. Ruppin. Co-evolving architectures for cellular machines. *Physica D*, 99:428–441, 1997.

[121] M. Sipper, M. Tomassini, and M. Capcarrere. Evolving asynchronous and scalable non-uniform cellular automata. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*, pages 382–387. Springer Verlag, 1997.

[122] A. R. Smith. Simple computation-universal cellular spaces. *Journal of the ACM*, 18(3):339–353, 1971.

[123] R. K. Squier and K. Steiglitz. Programmable parallel arithmetic in cellular automata using a particle model. *Complex Systems*, 8:311–323, 1994.

[124] D. Stevenson. Evolving cellular automata with genetic algorithms : analyzing asynchronous updates and small world topologies. Master's thesis, Portland State University, Portland, Oregon, 2008.

[125] R. Subrata and A. Y. Zomaya. Evolving cellular automata for location management in mobile computing networks. In *IEEE Transactions on Parallel Distributed Systems*, volume 14(1), pages 13–26, Piscataway, NJ, USA, 2003. IEEE Press.

[126] S. K. Tan and S.-U. Guan. Evolving cellular automata to generate nonlinear sequences with desirable properties. *Applied Soft Computing*, 7 (3):1131–1134, 2007.

[127] M. Tomassini, M. Giacobini, and C. Darabos. *Parallel Problem Solving from Nature - PPSN VIII*, chapter Evolution of Small-World Networks of Automata for Computation, pages 672–681. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004.

[128] M. Tomassini, M. Giacobini, and Ch. Darabos. Evolution and dynamics of small-world cellular automata. *Complex Systems*, 15(4):261–284, 2005.

[129] M. Tomassini and M. Venzi. Evolution of asynchronous cellular automata for the density task. In J. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, editors, *Parallel Problem Solving from Nature PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 934–943. Springer Berlin / Heidelberg, 2002.

[130] G. Y. Vichniac, P. Tamayo, and H. Hartman. Annealed and quenched inhomogeneous cellular automata. *Journal of Statistical Physics*, 45:875–883, 1986.

[131] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966. (edited and completed by A. W. Burks).

[132] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector and E. D. Goodman, editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 702–709. Morgan Kaufmann, 2001.

[133] J. Werfel, M. Mitchell, and J. P. Crutchfield. Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, 4 (4):388–393, 2000.

[134] P. R. Wiegand and J. Sarma. Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. *Parallel Problem Solving from Nature*, 1:912–921, 2004.

[135] J. Williams and M. Mitchell. Investigating the success of spatial coevolution. In *Proceedings of the 2005 conference on Genetic And Evolutionary Computation*, pages 523–530, Washington DC, 2005.

[136] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10D:1, 1984.

[137] S. Wolfram. Twenty problems in the theory of cellular automata. *Physica Scripta*, T9:170–183, 1985.

[138] S. Wolfram. *Theory and Application of Cellular Automata*. World Scientific Publishing, 1986.

[139] S. Wolfram. *A New Kind of Science*. Wolfram Media, Inc., Champaign, IL, 2002.

[140] D. Wolz and P. P. B. de Oliveira. Very effective evolutionary techniques for searching cellular automata rule spaces. *Journal of Cellular Automata*, 3(4):289–312, 2008.

[141] V. Zhirnov, R. Cavin, G. Lemming, and K. Galatsis. An assessment of integrated digital cellular automata architectures. *Computer*, 41(1):38–44, 2008.

APPENDIX A: GA EVOLVED RULES

A detailed explanation of the bit encoding of the rules listed in this appendix can be found in Chapter 2.

2DGKL rule
00000000000000000000000011111111001100110011001100000000111111 11000000000000000000000011111111001100110011001100000000111111 11110000000000000000111111111111111100110011001100111111111111 11111100000000000000001111111111111111001100110011001111111111 11111111000000000000000000000000111111110011001100110011000000 00111111110000000000000000000000001111111100110011001100110000 00001111111100000000000000001111111111111111001100110011001111 11111111111111000000000000000011111111111111110011001100110011 1111111111111111
"Naïve" rule (also called the *Local Majority* rule)
00000000000000000000000000000000100000000000000010000000100010 11000000000000001000000010001011100000001000101110001011101111 11110000000000000010000000100010111000000010001011100010111011 11111100000001000101110001011101111111000101110111111101111111 11111111000000000000000100000001000101110000000100010111000101 11011111110000000100010111000101110111111100010111011111110111 11111111111100000001000101110001011101111111000101110111111101 11111111111111000101110111111101111111111111110111111111111111 1111111111111111

Table A.1: Human design rules for 2D Density Classification Task (Moore neighborhood)

Cenek rule
00010000000110000011000100010101000100010110111001000000001100 11010100101001011100010111011010010000011101111001000001010101 01110100000000001000000001010000010100100001010000110000010100 00011100000000010111110000100101010011001011111111111101101111 11110101000000100000010000010100001101110001000101010101000000 11011111111000100110011111010000110111010100010111001111110101 11111101111100000001000001110011001100001111010011010101011101 00001101111111000100011011010101011001001101110111111101111111 0101111111111111
Marques-Pita rule (reported as rule 320 in [80])
00010001000100010000000000000000000100010001000100000000000000 00000100011111111100000000000000000001000111111111000000000000 00000001000111111111000000110000001100010001111111111111111111 11111100010001111111111000000110000001100010001111111111111111 11111111000100010001000100000011000000110001000100010001111111 11111111110001000111111111000000110000001100010001111111111111 11111111111100010001111111110000001100000011000100011111111111 11111111111111000100011111111100000011000000110001000111111111 1111111111111
Wolz and deOliveira rule (reported as rule 1 in [140])
00000000000000000000000000000001000000000001000100000001010000 01000010110001010100010100000001010000110100101011011101110101 01110000000100000001000100110000011100000011000001010100010101 01011100000000100101010101011111010111000000111001010101111111 11110111000010110001101101000101010101010000010100010011000100 11010001010001011101111111010001110101111100000111011111110111 11110111011111111111010101110100011101011101000111110111111101 10011101100101101111110011111100111111101111111111111111111111 0111111101110111

Table A.2: GA evolved rules for the 2D Density Classification Task (Moore neighborhood)

Global Synchronization
11111110111010111111110000111111111111101101011011111010011011 11111110011000111011101111011100101011000010111111000010001010 00001111110111000110111110100100111111111010111110101010101010 10000000111100101111011111010010000000111000011101100011100000 11100000111111111111111011010001010101001111011110101101001011 0 10110000001011101011111011101000101000010011011001100010001101 00000000000010101110101010001100110110110000101110001010010010 00001000001000011010100101100010010000100000000010000001100000 1010000010001000

Table A.3: The best performing GA-evolved rules for the two-dimensional Global Synchronization task.

Spatial Density Niching
00000000000100000000000000110111001000100100011100100111000101 11000010000000000100010000001100110100001000110001000001011011 10110000001110111110001110000001010000000010000001111011001001 11111101010000101110010111001101110011001111011011100110110011 11110011000100001000100000010011000000010000000110100001011000 10111001010000100010100011010011010001101000100010111100111011 10111110111100011010110101010110110110111000100111110101111 11111111111111000100000000101000010001110101110000000111111111 1111111111111111

Table A.4: The best performing GA-evolved rules for the two-dimensional Spatial Density Niching task.

Rectangle Image Bounding - Default Domain and Dense Variant Rule
00000010001000111010101101000100011100111110010111010111001111 00000110110010011011100010111100111101111011001111000011011001 00110111100010111000111001111110111010000001011100111100111011 11110111100101101111100100100111111101110111001101010011111110 00111001010101001001001111011011011100100000110011110110001011 11001010110001001011111011111000100011000101110100110010000111 01100101001100100110011011100111101111011110111011111101110110 11101101000111011010110100011110100100111101110010001001101011 1100101101110111

Table A.5: The best performing GA-evolved rules for the two-dimensional Rectangle Image Bounding task - default domain rule.

Rectangle Image Bounding - Sparse Variant Rule
00000100000000000000000000011101100110000001100110100001111001 1 11000000000000011000011001000101010000010100000111000001011111 11110100000000000100000010101111000101100000100001110010000100 01001100010000000100100110101001111111010001011110011101110111 11111111000000100001001100010001000101110000010101010111011100 11011000010110010000010001000011100011101110010010010011110110 01111111111101101011010111100011101110111011111101101001111100 10101111111111000000010101111101100111111111110101001111011111 1111111111111111

Table A.6: The best performing GA-evolved rules for the two-dimensional Rectangle Image Bounding task - spare variant rule.

APPENDIX B: MATHEMATICAL DEFINITIONS FOR STATISTICAL BASED FILTERS

B.1 LOCAL SENSITIVITY: MATH DEFINITIONS

Definition B.1.1 (Future light-cone layer of a site $(\vec{\eta}, t)$)**.** *Let $l^+(\vec{\eta}, t, \delta)$ be the sites in a CA space-time diagram, the state of which depends on $(\vec{\eta}, t)$, in the time-step $t + \delta$. The output of $l^+(\vec{\eta}, t, \delta)$ is a list of the sites in the CA's space-time diagram. These sites are determined according to the topology of the local neighborhood considered by each specific CA rule, and the value of $t + \delta$. For typical CA neighborhoods in 1D and 2D with radius $r = 1$, a depiction of the first three future light-cone layers is shown in Figure 7.7.*

Definition B.1.2 (Number of sites in a future light-cone layer for 1D CA with radius r)**.** *Given a future light-cone layer $l^+(\vec{\eta}, t, \delta)$ for a one-dimensional CA with neighborhood radius r, the number of sites in it can be determined from the expression,*

$$\Gamma(\vec{\eta}, t, \delta, r) = 2\delta r + 1$$

Definition B.1.3 (Number of sites in a future light-cone layer for von-Neumann 2D CA with radius r)**.** *Given a future light-cone layer $l^+(\vec{\eta}, t, \delta)$ for a two-dimensional CA with von-Neumann neighborhood radius r, the number of sites in it can be determined from the expression,*

$$\Gamma(\vec{\eta}, t, \delta, r) = (\delta r + 1)^2 + (\delta r)^2$$

Definition B.1.4 (Number of sites in a future light-cone layer for Moore 2D CA with radius r)**.** *Given a future light-cone layer $l^+(\vec{\eta}, t, \delta)$ for a two-dimensional CA*

with Moore neighborhood radius r, the number of sites in it can be determined from the expression,

$$\Gamma(\vec{\eta}, t, \delta, r) = (2\delta r + 1)^2$$

Definition B.1.5 (Future light-cone of $(\vec{\eta}, t)$ at depth d)**.** *Let $l^+(\vec{\eta}, t)$ be the set containing the union of all future light-cone layers $l^+(\vec{\eta}, t, \delta)$, in the range $\delta = \{1, ..., d\}$.* [1]

Definition B.1.6 (Difference plume between two light-cones)**.** *The difference plume between the original light-cone $l^+(\vec{\eta}, t)$ and a perturbed future light-cone $l^+(\vec{\eta'}, t)$, is,*

$$\Delta\left(l^+(\vec{\eta}, t), l^+(\vec{\eta'}, t)\right) = \frac{H\left(l^+(\vec{\eta}, t), l^+(\vec{\eta'}, t)\right)}{\sum_{\delta=1}^{d} \Gamma(\vec{\eta}, t, \delta, r)}$$

where the function H is the Hamming distance between the two future light-cones that have the same topology and size, i.e.

$$\Gamma(\vec{\eta}, t, \delta, r) = \Gamma(\vec{\eta'}, t, \delta, r)$$

for a fixed radius r and for every value of $\delta = 1, 2, ..., d$.

Definition B.1.7 (Local Sensitivity of site $\vec{\eta_o}, t)$ with future-depth d and perturbation-range p)**.**

$$\xi_d^p(\vec{\eta}, t) = \frac{\sum_{i=1}^{|S|} \Delta\left(l^+(\vec{\eta}, t), l^+(\vec{\eta^i}, t)\right)}{|S|}$$

where $l^+(\vec{\eta^i}, t)$ corresponds to the i^{th} future light-cone resulting from replacing the original perturbation neighborhood with $s_i \in S$, and $l^+(\vec{\eta}, t)$ to the original future light-cone for $(\vec{\eta}, t)$.

[1] *A notation for referring to a future light-cone using a subscript (d) that denotes the specific value of the future depth, i.e. $l^+_{(d)}(\vec{\eta}, t)$ could be used as well, but this will be omitted here for notational simplicity, under the assumption that a light-cone can only exist for a specific finite value of d.*

B.2 LOCAL STATISTICAL COMPLEXITY: MATH DEFINITIONS

More formally, the computation of the local statistical complexity for a site in a CA space-time diagram, $\mathcal{C}(\vec{\eta}, t)$, requires the following mathematical definitions:

Definition B.2.1 (Observed past and future light-cone configurations for a specific CA rule)**.** *Let the sets L^- and L^+ denote, respectively, the collection of (randomly ordered) distinct past light-cone and future light-cone configurations observed in the space-time diagram. A member of either set L^- (or L^+) will be denoted by l_i^- (or l_i^+), where i corresponds to the position of the specific past (or future) light-cone in the source set.*

Definition B.2.2 (Estimated conditional distribution matrix of observed past and future light-cones)**.** *Let matrix M represent the conditional frequency distributions of the observed past and future light-cone configurations. The row headers represent a set of all unique past light-cone configurations L^-, and the column headers are all unique future light-cone configurations L^+ for a given CA space-time diagram. A specific matrix element has a value of:*

$$m_{i,j} = \sum_{all\ \ sites} (l_i^- | l_j^+)$$

where $m_{i,j}$, corresponds to the number of times the past light-cone l_i^- has been followed by the future light-cone configuration l_j^+.

Definition B.2.3 (Estimated conditional distribution vectors)**.** *Let the m_i^- denote the i^{th} row vector of matrix M that represents the frequency of observed future light-cones L^+ given a past light-cone l_i^-. This means that the notation m_i^- is simply a more compact notation of $P(L^+|l_i^-)$.*

Similarly, let the m_j^+ denote the j^{th} column vector of matrix M representing the estimated conditional distribution of $P(L^-|l_j^+)$.

Definition B.2.4 (Similarity between a pair of conditional distribution vectors)**.** *The similarity function S between two estimated conditional distribution vectors m_i^- and m_k^- is defined as:*

$$\mathcal{S}(m_i^-, m_k^-, \alpha) = \begin{cases} 1 & \chi^2(m_i^-, m_k^-) < \alpha \\ 0 & otherwise \end{cases}$$

where α is a similarity threshold constant, the value of which is established empirically. The value of $\chi^2(m_i^-, m_k^-)$ is determined by calculating the following:

$$\chi^2(m_i^-, m_k^-) = \sum_{j=0}^{|L^+|} \frac{(m_{i,j} - m_{k,j})^2}{m_{k,j}} \quad where \;\; (m_{i,j} - m_{k,j}) \neq 0, \;\; m_{k,j} \neq 0$$

Definition B.2.5 (Equivalence class $\epsilon_{m_i^-}$)**.** *The equivalence class, $\epsilon_{m_i^-}$, is a set of all estimated conditional distribution vectors m_λ^- that do not belong to any other equivalence class and the similarity $S(m_i^-, m_\lambda^-, \alpha) = 1$ (where m_i^- is the conditional distribution vector representing the equivalence class $\epsilon_{m_i^-}$, and m_λ^- is the candidate conditional distribution vector). The equivalence class $\epsilon_{m_i^-}$ of similar conditional distribution vectors is defined as following:*

$$\epsilon_{m_i^-} = \{m_\lambda^- : \mathcal{S}(m_i^-, m_\lambda^-, \alpha) = 1\}$$

Definition B.2.6 (Set of equivalence classes ϵ_M)**.** *The set of equivalence classes, ϵ_M, for a given estimated conditional distribution matrix M is a set of disjoint subsets $\epsilon_{m_i^-} \in M$:*

$$\epsilon_M = \bigcup_i \epsilon_{m_i^-}$$

where $\epsilon_{m_i^-} \cap \epsilon_{m_j^-} = \emptyset, \forall(\epsilon_{m_i^-}, \epsilon_{m_j^-}) \subset \epsilon_M : i \neq j$.

Definition B.2.7 (Probability of an equivalence class $\epsilon_{m_i^-}$). *The probability that an arbitrary past light-cone is a member of an equivalence class $\epsilon_{m_i^-}$ is given by the formula,*

$$Pr(\epsilon_{m_i^-}) = \frac{\sum_{k} \sum_{j=1}^{|L^+|} m_{k,j}}{\sum_{i=1}^{|L^-|} \sum_{j=1}^{|L^+|} m_{i,j}} \; where \; k : m_k^- \in \epsilon_{m_i^-}$$

The numerator is a sum of the frequency counts of the conditional distributions that belong to the equivalence class $\epsilon_{m_i^-}$, and the denominator is the number of all analyzed sites in the space-time diagram.

Definition B.2.8 (Local Statistical Complexity $\mathcal{C}(\vec{\eta}, t)$). *The Local Statistical Complexity associated with a site $\vec{\eta}$ at time t in the space-time diagram of a CA is $-log_2$ of the probability associated with the equivalence class to which the past light-cone of the site $(\vec{\eta}, t)$ belongs to:*

$$\mathcal{C}(\vec{\eta}, t) = -log_2(Pr(\epsilon_{m_i^-})) \; where \; l^-(\vec{\eta}, t) \in \epsilon_{m_i^-} : \epsilon_{m_i^-} \in \epsilon_M$$

B.3 LOCAL INFORMATION STORAGE, INFORMATION TRANSFER AND INFORMATION MODIFICATION: MATH DEFINITIONS

Definition B.3.1 (Basic definitions: current site, past configuration, and neighboring sites.). *Let the expression $x_{i,n+1}$ refer to the state of the automaton's i^{th} site at time $(n+1)$, also called the current site.*

The past vector $\langle x_{i,n}^{(k)} \rangle$ denotes k previous states of site i from the time n to $n-k$.

$$\langle x_{i,n}^{(k)} \rangle = \langle x_{i,n}, x_{i,n-1}, x_{i,n-2}, \ldots, x_{i,n-k} \rangle$$

The previous configuration of l many left (or right) adjacent cells to the current site are denoted by vector $\langle x^{(l)}_{i-j,n} \rangle$ *(or* $\langle x^{(l)}_{i+j,n} \rangle$*). In Moore neighborhood two-dimensional CA, j refers to the eight neighboring sites adjacent to the current site and four neighbors for van Neumann neighborhood. An abbreviated shorthand for all neighboring sites is* $\langle x^{(l)}_{i \pm j,n} \rangle$

Definition B.3.2 (Frequency vectors)**.** *The composite pattern is defined as a concatenation of one or more structures defined in B.3.1. A frequency vector stores the occurrence counts of a particular composite pattern. For a two state CA, each binary composite pattern encountered is encoded as an offset to the frequency vector where the counter is incremented (the length of the frequency vectors is* $2^{patternlength}$*). The pattern statistics are recorded from "several" space-time diagrams with random initial configurations. The frequency vectors and the composite pattern definitions are listed below:*

Vector Name	*Pattern Definition*	*Pattern Length (bits)*	*Vector Description*
P	$\langle x_{i,n}^{(k)} \rangle$	k	*k previous configurations of the current site*
C	$\langle x_{i,n}^{(k)}, x_{i,n+1} \rangle$	$k+1$	*k previous configurations of the current site* with *the state of the current state*
S	$\langle x_{i,n+1} \rangle$	*1*	*the current site*
PL	$\langle x_{i,n}^{(k)}, x_{i-j,n}^{(l)} \rangle$	$k+l$	*k previous configurations of the current site* with *l left neighboring sites*
CL	$\langle x_{i,n}^{(k)}, x_{i-j,n}^{(l)}, x_{i,n+1} \rangle$	$k+l+1$	*k previous configurations of the current site* with *l left neighboring sites* with *the state of the current site*
PR	$\langle x_{i,n}^{(k)}, x_{i+j,n}^{(l)} \rangle$	$k+l$	*k previous configurations of the current site* with *l right neighboring sites*
CR	$\langle x_{i,n}^{(k)}, x_{i+j,n}^{(l)}, x_{i,n+1} \rangle$	$k+l+1$	*k previous configurations of the current site* with *l right neighboring sites* with *the state of the current site*

Definition B.3.3 (Local Information Storage)**.** *The Local Information Storage a in site i at time $n+1$ is defined as:*

$$a\left(i, n+1\right) = \lim_{k \to \infty} log_2 \frac{p\left(x_{i,n}^{(k)}, x_{i,n+1}\right)}{p(x_{i,n}^{(k)}) p\left(x_{i,n+1}\right)} = log_2 (\frac{\overline{c_i}}{\overline{p_i} \times \overline{s_i}})$$

Definition B.3.4 (Local Information Transfer)**.** *The Left Local Information Transfer is defined as a faction of following conditional probabilities:*

$$t_{left}\left(i,n+1\right)=\lim_{k\to\infty}log_2\frac{p\left(x_{i,n+1}|x_{i,n}^{(k)},x_{i-j,n}^{(l)}\right)}{p\left(x_{i,n+1}|x_{i,n}^{(k)}\right)}=log_2\frac{\overline{cl_i}\div\overline{pl_i}}{\overline{c_i}\div\overline{p_i}}$$

Right Local Information Transfer is defined analogously as following:

$$t_{right}\left(i,n+1\right)=\lim_{k\to\infty}log_2\frac{p\left(x_{i,n+1}|x_{i,n}^{(k)},x_{i+j,n}^{(l)}\right)}{p\left(x_{i,n+1}|x_{i,n}^{(k)}\right)}=log_2\frac{\overline{cr_i}\div\overline{pr_i}}{\overline{c_i}\div\overline{p_i}}$$

Definition B.3.5 (Local Information Modification)**.** *The Local Information Modification is a sum of Local Information Storage and the Local Information Transfer from all directions. The equation below calculates the information modification for one-dimensional CA:*

$$s\left(i,n+1\right)=a\left(i,n+1\right)+t_{left}\left(i,n+1\right)+t_{right}\left(i,n+1\right)$$