Computation in the Wild: Reconsidering Dynamic Systems in Light of Irregularity

Tony Liu

March 25, 2016

List of Figures

Abstract

Chapter 1

Introduction

Dynamic natural systems have the ability to perform complex tasks that resemble computation, often without the aid of a central processing unit. In particular, spatial systems that are only locally connected, such as plant cell arrays or single-agent populations, can produce emergent, globally-coordinated behavior [6, 29]. These systems are of particular interest because of their massive parallelism and fault tolerance within noisy, imperfect environments [38]. Decentralized dynamic systems can be described and modeled using cellular automata (CA) because of CA's ability to support complex behavior arising from simple components and strictly local connectivity [28]. The goal with CA models is to able to examine how and under what conditions global, distributed computation emerges in such systems.

One of the main motivating examples of emergent natural computation we have been examining is plant cell stomatal coordination. These dynamic pores control gas exchange within the plant and "solve" a constrained optimization problem without a central communication system. What is remarkable about stomata is that their behavior is statistically indistinguishable from the behavior of CAs that solve the *majority problem*: a global coordination task where all cells in the CA need to converge to the majority state of the initial configuration [29, 33, 39]. Furthermore, the corresponding majority task automata have been shown to respond robustly in the face of variations in their environment such as state noise, much like their biological counterparts [25]. This relationship is exciting because it has been shown that computation in CAs can only occur under specific conditions, existing at a *critical point* between ordered and chaotic behavior [21, 42]. Thus, perhaps we could begin to understand how computation emerges in nature given their established connection with CAs. However, the automata models built to study natural systems like plant stomata make a potentially limiting assumption: regularity of both the cellular grid and the local connections. Dynamic systems in nature certainly do not appear in a uniformly connected lattice. Like their surrounding environment, the spatial orientation of these systems are noisy and prone to irregularities. While some work has been done to see the effects of nonuniform grids such as Voronoi diagrams or Penrose tilings on automata performance [8, 15, 19], further study is needed to ensure that our models of computation can safely be "mapped" to the systems occurring in nature.

With these considerations in mind, this work would pursue a deeper exploration of irregular computing CAs. We plan on conducting a series of experiments on a spectrum of automata with irregular grids and connectivities, quantifying their behavior and comparing them to traditional

CAs. As noted earlier, computation appears to emerge at a point of criticality somewhere in between the ordered and chaotic automata and Langton's λ parameter, representing the relative order of a CA, is used to parametrize the possible automata space [21]. λ is important because it is an indicator of the class of CAs that have the ability to compute [43]. However, this measure is tuned specifically for static neighborhood definitions and uniform grids. With the experimentation on irregular automata, the goal is to explore and develop a more general notion of λ and other metrics so that we can better understand and perhaps even quantify the conditions in which computation can emerge in noisy systems. Another important question we hope to address with this work is whether there is a qualitative difference between regular and irregular grid patterns and connectivities: are there cases where uniform CA models are sufficient for representing biological systems?

We believe that the study of CA behavior in irregular environments is critical to achieve a greater understanding of how biological systems combat imperfections. Ultimately, the contribution of work on natural computational systems is twofold: not only can we achieve a better understanding of how some biological processes operate, but knowledge of how these systems work can inspire alternative computing methods [24, 37]. The hope is to illuminate how nature is able to perform complex computation in noisy environments and apply these lessons to advance future computing models.

We will begin with a review of previous work in Chapter 2, spanning the concepts of criticality, robustness, and spatial representations in computing dynamic systems and examining the potential applications of such models. We will describe our system architecture in Chapter 3 as well as the tools we built to explore complex behavior in CA systems.

Chapter 2

Previous Work

Why are we interested in distributed computational CA systems? The goal is to achieve a better understanding of how computation naturally emerges from such systems in order to accurately model biological distributed processes as well as improve our technological computing models. We will begin with some motivating examples by considerings previous work on applications of robust, distributed CA models, spanning topics such as plant biology, population dynamics, geographic modeling, and computer architecture. However, in order to effectively apply CA models of natural systems, we must understand what computation means in such dynamic systems. Are there requisite properties of computation, and if so how do they arise? For this we consider a collection of papers that develop a framework for quantifying computational characteristics as well as present a hypothesis that computation emerges at the "edge of chaos." Next, as we need to understand how natural computation is robust in the face of noisy and irregular environments, we will examine work on noise and fault tolerance in distributed automata systems. Finally, spatial irregularity is a prevalent feature in nature that is often overlooked in such studies, so we consider some work that examines the impact of such irregularity on dynamic CA systems.

2.1 Motivation and Applications

2.1.1 Cellular Automata for Biological Modeling: Stomatal Patchiness

An important motivating example for this work is the modeling of plant cell stomatal coordination done by Peak et al.. Plant stomata are dynamic pores distributed across leaves that control gas exchange via the opening and closing of their apertures. These stomata solve a constrained optimization problem, as they must maximize the uptake of CO_2 while minimizing water vapor loss [29, 39]. Long believed to be autonomous units that respond similarly but independently to environmental conditions, it has now been shown that a stoma's aperture is also dependent on interactions with neighboring stomata, sometimes producing coordinated behavior called *stomatal patchiness*, where large groups of stomata uniformly open or close [33]. This phenomenon is poorly understood by biologists, due to its apparent negative impact on gas exchange optimization as well as its highly variable behavior. However, with biological evidence that plant stomata interact locally [33], task-performing cellular automata are suitable

candidates for modelling patchy stomatal conductance. In fact, modeling has shown that the stomatal systems are statistically indistinguishable from CAs configured to solve the *majority task* or *consensus problem*, which involves determining which state (0 or 1, open or closed) is the majority state in the initial configuration and then converging all cells to that state (Figure 2.1) [17]. Qualitative comparisons yield similarities to these task-performing automata as well (Figure 2.2). The patchiness observed in the plant stomata are analogous to transient periods that are essential for the computation of majority in CAs. What is remarkable about these stomatal systems is their response to irregularity, as they must manage highly variable and imperfect environments. The corresponding majority task CAs have been shown to respond robustly in the face of state noise, imperfect information transfer, and transition rule heterogeneity; in fact, task performance is enhanced in some cases by the presence of noise [25].

It is important to note however that none of these systems are guaranteed to optimally solve the majority task, and even determining whether or not a single instance will perform optimally is hard. The inherent difficulty of determining the success or failure of a given task-performing network emphasizes the importance of information granularity; in an appeal to the concept of self-organizing criticality [5], small pieces of fine grain information about initial configurations may lend more insight to the global behavior of a distributed network than large but coarse bodies of information about the structure of the space. Thus, studying stomatal patchiness not only provides a step towards connecting computation to phenomenon in nature but also illustrate the central ideas of robustness and criticality.

2.1.2 CA Models of Social and Ecological Dynamics

Researchers in ecology and the social sciences often use CA models when modeling dynamics because automata models encapsulate essential features of many real-world processes. The discrete and distributed nature of CA system provide a tractable system where agents (represented by individual cells) can interact in local and overlapping neighborhoods [18, 6]. In particular, the local interconnections present in cellular automata models illuminate how micro-level rules can give rise to certain macro-level effects, a crucial aspect of domains such as population dynamics. Because the micro/macro relations are readily apparent in CA systems, scientists in these fields can utilize such models to not only produce quantitative simulations and predictions in but also to achieve a qualitative understanding of how the real world operates [18]. Cellular automata are appealing because they take a "simplistic" approach to "complex" modeling, which not only is useful in domains where details on the underlying mechanisms are not well understood (such as animal herd movement or forest fire behavior) but also reduces multi-agent systems to their fundamental one-on-one interactions [6, 8], again alluding to the importance of fine-grain information when trying to understand coarse-grain processes.

Scientists in these fields recognize the limitations of CA models as well, as there are concerns that the discrete and uniform nature of these systems are potentially prohibitive in providing a sufficiently accurate representation of the real-world phenomena they are trying to model. In particular, there is a need for variability in both the spatial construction of the modeling environment as well as the local relationship between cells [15] that the regular lattice structure of traditional cellular automata do not provide. We will consider these limitations as well as

work on alternative spatial configurations in more detail in Section 2.4.

2.1.3 Cellular Automata as Inspiration for Novel Computing Models

Additionally, distributed CA systems have provided inspiration for new forms of computing. Computing models based on cellular automata are founded upon three fundamental principles: simplicity, "vast parallelism", and locality [38]. Similar to the appeal of automata systems in dynamic modeling, CA-based computers utilize simple "cellular" computational units in tandem to solve complex computational tasks, can invoke parallelism on a scale that traditional parallel systems cannot achieve, and have better fault containment as well as more tolerance of imperfect input and execution due to strictly local connectivity.

An example of such a cellular-based computing is the CAM-8 architecture developed by Margolus. The CAM-8 computer architecture is motivated by the idea that in order to maximize computational density, the underlying structure of the computer must mimic the basic spatial locality of physical law [24]. Thus, the architecture is mesh network of local CA "compute nodes" as they accurately represent the local interconnectivity that is present in real-world micro-physical systems. As a result, there is a correspondence between how computation is performed under the architecture and the physical implementation of the system itself. Because of this adherence to physical law, the CAM-8 architecture is particularly capable of computing spatially moving data, ideal for particle simulation and medical imaging, among others [24]. The goal with CAM-8 and other distributed cellular computing models is not necessarily to replace serial computers, but rather to find particular domains of problems where these alternative computing models can excel in and perhaps surpass traditional computing models.

These cellular-based computing architectures are not without their own limitations. One of the main challenges in cellular computing is being able to find local interaction rules that express the overall problem that needs to be computed. There is no good way to abstract beyond the local CA rules to provide a high-level interface for programmers to develop in, making deliberate and controlled global behavior difficult to produce. Indeed, this is the main barrier CAM-8 must overcome with in order to become a viable and practical computer architecture. There is also an issue of scalability; scaling grid size does not necessarily bring about performance scaling or even task scaling: whether or not the same level of performance on the task is maintained [38]. Nevertheless, the potential applications in building cellular-based computational models illustrate the promise of examining robust distributed computation in natural CA-like systems.

2.2 Criticality and the Emergence of Computation

2.2.1 Foundations of Computation in Dynamical Systems

What are the requisite conditions for computation to emerge from a dynamical system? It has been shown that various CA systems can support universal computation [42], but what are the particular characteristics that allow for computation to be possible? nnComputational constructs such as Turing machines and other equivalent entities are built upon three fundamentals that can be formulated in the dynamics of a CA system. The system must be able to

support the *storage* of information, with the ability to preserve information for arbitrary time periods. Information *transmission* across arbitrarily long distances must also be possible in the environment. Finally, there must be some mechanism for information *interaction* with the potential for information to be transformed or modified [21]. These properties are necessary for any dynamical system, automata or otherwise, to have the capacity for computation, but are not sufficient to give rise to computation by their presence alone. Langton also establishes the notion of dynamical systems undergoing "physical" phase transitions between highly ordered to highly disordered dynamics, with the most interesting behavior occurring within the boundaries of this transition. This transition region is also where the three requisite properties of computation often occur. Thus, the hypothesis Langton claimed is that computation can spontaneously emerge and dominate the dynamics of a physical system when such a system is at or near such a "critical transition" point [21].

2.2.2 Metrics

Throughout the pursuit of understanding computation in dynamical CA systems, many metrics have been utilized and created that can help quantify particular computational characteristics. We will touch on and present some of them here, with equations listed in Appendix A.

We will begin with a formal definition of a cellular automaton [21]. A CA is composed of a lattice of dimension D with a finite automaton present in each cell of the lattice. There is a finite neighborhood region $\mathcal{N}$, where $N = |\mathcal{N}|$ is the number of cells covered in the neighborhood region. Typical neighborhood stencils for two-dimensions are the five-cell *von Neumann neighborhood* and the nine-cell *Moore neighborhood* (Figure 2.3). Each cell contains a finite set of cell states Σ of size $K = |\Sigma|$ and a transition function Δ, which maps a set of neighborhood configurations to the cell states: $\Delta : \Sigma^N \to \Sigma$. We typically characterize a particular class of cellular automata by the number of neighbors N and the number of cell states K.

Wolfram proposed a qualitative method for classifying CA automata behavior, with systems falling into one of four classes [28, 41]:

- Class I: All initial configurations relax to the same fixed, homogeneous state.
- Class II: The CA relaxes to simple periodic structures, perhaps dependent on the initial configuration.
- Class III: Most initial configurations degenerate to chaotic, unpredictable behavior.
- Class IV: Some initial configurations result in complex localized structures that have the potential to be long-lived.

Li et al. later expands to six categories, with Class I and Class II both split into more specific subclasses. Though these are broad, rough categorizations, we expect Class IV to be the category of interest when examining potential capacity for computation in cellular automata.

The parameter λ (Appendix A.1) as established by Langton is used to both narrow the space of CAs to consider as well as to measure the relative homogeneity or heterogeneity of a CA rule table: a completely homogeneous rule table maps all entries to a single "quiescent" state whereas a completely heterogeneous rule table maps entries to random states [21]. λ is the fraction of the number of non-quiescent mappings in a given rule table, and can be thought

of the "average" amount of order a given automata transition rule set possesses. Thus, λ values range from 0, which represents a completely homogeneous rule table to $1 - \frac{1}{K}$, which represents a completely heterogeneous table. There is also a notion of "critical" λ, denoted λ_c, where the most complex dynamics tend to emerge. λ_c and idea of criticality will be examined in more detail in Section 2.2.3. The hypothesized relationship between λ and Wolfram's complexity classes can be seen in Figure 2.4.

The statistical quantity γ (Appendix A.2) developed by Li et al. is another metric that describes the average asymptotic motion or spreading rate of the *difference pattern* in a CA. The difference pattern is a measurement of how two different configurations on a automata lattice become either more different or more similar when applying transitions from the same rule table [23]. γ can roughly be seen as the "variance" to λ's "mean" when considering the relative order or chaos, as γ provides finer-grain information about the dynamics of particular CA that may not be distinguishable when only considering λ.

A common quantity used to measure the relative order in a CA system is *Shannon entropy*, denoted by H (Appendix A.3) [21, 22, 43]. We can think of Shannon entropy as measuring the amount of information present in the CA space based on the frequency of cell state occurrences; there is less information in ordered systems and more in disordered systems. Thus a completely homogeneous rule table will yield an entropy of $H = 0$ while a completely heterogeneous rule table will yield the maximum entropy possible for that particular (N, K) class of automata. A natural extension to this measure is *mutual information* (A.4), defined as the correlation between two individual cell entropies [21]. We expect some amount of mutual information being shared across cells in order for computation to be supported; too little shared information degenerates to chaotic behavior, while too much mutual information creates highly correlated structures that are too rigid to support computation.

Mean field theory from the description of many-body systems in physics is often used to approximate a number of the metrics described above [23, 43]. The idea is to quantify a many-body system not by considering all mutual two-body interactions (which may become intractable), but rather to describe the interaction of one particle with the the rest of the system by an "average potential" created by the other particles. This approximation technique is particularly useful when considering classes of automata in the limit, such as in the analysis performed by Wootters and Langton. Approximations for λ_c, γ, and H using mean field theory can be found in Appendix A.5.

2.2.3 The Edge of Chaos: Investigating where Computation Emerges

Langton introduces the idea that computation emerges from "the edge of chaos," the critical transition region that separates ordered and chaotic behavior [21]. His primary method of investigation is a Monte Carlo sampling of two-dimensional CAs, comparing the relationship between the parameter λ and the average dynamical behavior of the system, measured by entropy. From both qualitative and quantitative analyses, the most interesting action in 2D CAs occurs in middling λ values, where the *transient lengths* before static structures emerge is arbitrarily long. Low levels of λ have short transient lengths before the CA crystallizes (Figure

2.5), and high levels of λ have short transient lengths before the CA degenerates into chaos (Figure 2.6). Levels of λ in the middle however are a "sweet spot" of entropy (Figure 2.7); since information storage involves lowering entropy while transmission involves raising entropy, a balance must be struck in the overall system to support these foundations of computation. Mutual information is another measure that captures this same balance: if the correlation of entropy between two given cells is too high, they are overly dependent and have a tendency to crystallize, but if the correlation is too low, the cells are effectively acting independently, indicating chaos.

Dynamical systems such as CAs exhibit characteristics that echo to the decideability of computation. CAs that live below the transition point quickly crystallize or "freeze," while CAs above the point degenerate into random chaos. The behavior of dynamical systems at the fringes of the λ spectrum can thus be determined, while CAs that live near the transition point cannot; this *freezing problem* shows that it is undecideable whether or not a particular CA rule set within this complex region will either crystallize or degenerate due to the long transient lengths. Since computers are simply highly formalized dynamical systems, the classic Halting Problem can be viewed as a specific instance of the freezing problem. Through these experiments and observations on the relationship between λ and dynamic, Langton has established a bound on the complexity of dynamical systems; systems located near the edge of chaos exhibit a wide range of complex behavior that contain the foundations of computation. Thus, there is a narrow location in the λ spectrum of dynamical systems where emergent computation can be discovered, with the maximum complexity occurring at λ_c (Figure 2.8).

Further work attempted to pinpoint precisely where the transition point occurs in a class of CA behavior, perhaps continuing along the parallels between CAs and transitions in physical systems. Utilizing mean-field theory, theoretical approximations of the entropy against λ in two-dimensional automata made by Wootters and Langton closely match previous experimental results. As the number of total possible cell states approaches ∞, it has been determined that there is a sharp phase transition occurring at $\lambda = 0.27$, akin to first-order transitions in actual physical systems [43]. Perhaps this is the λ_c point for this particular class of CAs. These results show promise in the parallels between physical systems and the dynamic behavior of cellular automata. However, to investigate this in further detail, it may be necessary to identify other statistical properties in order to fully determine the phase transition point.

Likewise, Li et al. examined the dynamics of CA behavior with respect to γ in an effort to better characterize the transition region. They determined that this region between ordered and chaotic behavior is not smooth, but rather a complicated structure in of itself as values of γ fluctuate widely within the space [23]. Most of the transition region is a boundary between ordered and chaotic, with abrupt jumps in behavior for CAs that cross this border. However, other areas of the region appear to have a "thickness" in the boundary, yielding smooth changes in statistical measures through this critical subregion (Figure 2.9). Thus Li et al. also come to the conclusion that λ alone cannot specify the critical region for dynamic CA systems.

Others have questioned whether a precise critical point where computation emerges truly exists. Mitchell et al. reexamine the relationship between λ and the dynamic behavior of ordered and unordered CAs by attempting to replicate experiments that suggest the existence

of "critical" λ_c in previous work by Packard. Though Wootters and Langton determined that there is a convergence to a critical point in the limit of infinite state CAs, the relationship between λ and critical regions is less clear for finite state CAs. What is important to note is that the variability of CAs at a given lambda value could be high (measured by the difference pattern spreading rate, γ), and so the behavior of a particular rule at a given value of λ might be very different from the average behavior at that value. Packard utilizes a *genetic algorithm* to evolve one-dimensional, two-state automata ($K = 2, N = 3$) that solve the majority task; his results indicate clustering around λ_c in the final population of CAs, which is cited as evidence for complex behavior emerging at the boundary points. However a theoretical examination shows that, given the natural symmetry in the majority task problem, the optimal λ value occurs at 0.5, not at the critical λ point. Mitchell et al. then run their own genetic algorithm, showing that the best CA rules in fact cluster around 0.5. It is important to note that for that particular class of one-dimensional, two-state automata, $\lambda = 0.5$ is considered to be in the region of "chaotic" CAs [27].

These results illustrate an important point: statistics that measure the behavior of these CAs may apply an implicit "filtering bias" on the class of automata, and so structure may be revealed under different measures that is not present under other measures, as shown by λ's classification of the optimal CAs for the majority task as "chaotic." Langton's hypothesis that λ correlates with computational ability appears to need refinement: λ may very well correlated with the average behavior of CAs but what is the significance of "average behavior," especially when γ is high in the critical regions of interest? For a given N and K, CAs that have the potential to compute may reside in a wide range of λ values, illustrating the need for a more precise metric. Mitchell et al. also stress that the focus should be harnessing the computation "naturally" present within a CA rather than attempting to impose traditional definitions of computation upon them.

Nevertheless, the relationship between critical regions and computation in general remains important and needs to be closely examined at a finer grain. Instead of considering the average behavior of automata, Crutchfield and Hanson decompose and filter the spatial configurations of one-dimensional CAs as they move through time. The central goal of their work is to distill a seemingly "turbulent" system by finding a *pattern basis* representation of the CA space that will serve as the backdrop for potential coherent structures present in the system to be identified [10]. These pattern bases form "regular domains" that are *temporally invariant*, where the CA rule table maps configurations within a regular domain to the same regular domain [28]. With these stable domains identified, they can be filtered out of the space-time diagrams, with the dynamics of interest becoming apparent at the boundaries between regular domains, which are particle-like in behavior [10]. Thus, the dynamics of a CA can be examined in this manner to illuminate underlying structure, providing more useful and finer-grained information than simply classifying the system as ordered or chaotic as shown in Figure 2.10. Crutchfield and Hanson calls these domains and structures the "intrinsic computation" of the CA, as they can be understood in computational terms regardless of the global computation taking place in the CA [28]. The intrinsic computation in itself is an interesting unit of analysis, capable of generating rich interactions at the domain borders that may in fact cause coordinated behavior

to emerge [11]. Again, this stresses the importance of information granularity, as the presence of many smaller regular domains that have inherently different structures in the CA space can cause statistics collected across the global domain to be misleading [10].

Throughout our exploration of computation in distributed dynamic systems, we often see a delicate balance struck between order and chaos in order to support complex behavior: information must be both propagated (high entropy) and stored (low entropy), and minute alterations in local configurations can have catalytic global effects. What happens when this balance is disturbed? Thus far we have only considered highly regular environments for computation in CA systems, with uniform lattice grids and rule tables. Do these systems behave robustly when the environments are not well-structured? Dynamic natural systems serve as "the ultimate proof of concept," as they are able to function despite residing in environments (the real world) that are inherently noisy and irregular [38]. We'll look first to inspiration from nature when considering fault tolerance in computing CA systems.

2.3 Robustness in Face of Irregularity

2.3.1 Stomatal Dynamics Revisited: Utilizing Environmental Noise

We begin our review of robustness by revisiting plant stomatal dynamics. Unlike technological systems, biological systems such as stomatal arrays on plant leaves are often able to manage variable environments "innately," without external intervention or control. Indeed, though stomatal systems behave similarly to majority task CAs as noted earlier in Section 2.1.1, they face a number of irregularities not accounted for in the CA models such as *heterogeneous interactions*, where local interactions may not be uniform due to stomatal size, orientation, and spacing, or *modularity*, where leaf veins cause stomata to be subdivided into modules that interact at a level beyond individual stoma interaction [25]. Additionally, the stomatal systems must handle temporal *state noise*, where there is unavoidable variability and imprecision in the stomatal functioning itself across time. Thus, Messinger et al. examine the behavior and performance of stomatal-inspired majority task CAs in environments designed to mimic the irregularities plant stomata have to contend with.

When considering the majority task, most initial configurations are relatively easy for CAs to classify correctly, where there are a large proportion of either "on" or "off" cells present in the space. Unsurprisingly, the most difficult cases are when the density of on and off cells are roughly equivalent. Messinger et al. note that these hard cases often cause patchiness and are solved correctly only when the patches travel coherently across the CA space, emphasizing the importance of information transfer when supporting task performance in these locally connected networks. Unsuccessful CAs often have "stuck" patches that are unable to be resolved. Irregularities in the form of heterogeneous interactions, simulated by nonuniform rule tables and random connectivity, and modularity, simulated by grafting together "mosaic" networks of modules, both reduce the performance of the task CAs but do not cripple them [25]. Perhaps most surprisingly, the majority task CAs actually *improve* in performance when small amounts of state noise are introduced to the system while other defects are present. Messinger et al. hypothesize that this is because the state noise cause small perturbations in the system, stim-

ulating movement and perhaps freeing previously stuck patches to move about the space once more. We have seen how small changes can snowball and sweep through entire the entire CA space, and it appears that natural systems embrace noise and use it to their advantage. Indeed, randomness appears essential in facilitating the self-sustaining and self-informing function of decentralized biological systems [26]. We will turn to Mitchell's analysis of how these systems might operate to illustrate how such structures can maintain themselves in such complex environments despite the lack of central control.

2.3.2 Redundancy in Decentralized Systems

Mitchell presents two instances of decentralized systems, the human immune system and ant colonies, which perform complex tasks via "adaptive self-awareness:" global information about the system dynamically shapes and feeds back to the movements and actions of the lower level components. What is interesting about these systems is that single agents act naively, tending to conform to what the dominant behavior is locally: for example, an ant in an ant colony will have an increased probability of switching to and working on a task if it observes many other ants in its immediate environment working on it [26]. However, it is the probabilistic aspect of these decisions, the idea of sampling a small "neighborhood," that allow the complex behavior to emerge. Similar to an ant's local sampling, a lymphocyte will bond with molecules indicative of pathogens it encounters locally and will reproduce proportional to the strength of the bond it forms, facilitating a Darwinian process in that the best lymphocytes for combating the pathogen will emerge on a system-wide scale.

From studying immune systems and ant colonies, Mitchell emphasizes the need for randomness and probabilistic decisions in order for controlled behavior to occur: these systems are not just robust to variations in their environment, but actually require irregularities and noise to function properly. Additionally, they illustrate the importance of redundancy and sampling. A single lymphocyte or ant is a fragile unit, and so the global system cannot be dependent on any specific agent present within its system. Instead, there is an inherent redundancy within the system, with many agents sampling their local environments, yielding independent, fine-grained behavior that only become significant when considered globally [26]. Thus, the individual components do not rely on each other making the system robust to faults, yet the overall behavior is highly coordinated.

We see the ideas of redundancy and random noise come into play again in Ackley and Small's efforts to mimic natural robustness to variability in their *Moveable Feast Machine,* a dynamic CA-based architecture designed with the hope of producing "indefinitely scalable," robust computing. The computational entities, called *atoms*, move about probabilistically in a two dimensional grid, independently and asynchronously acting on their local neighborhood. A sorting routine implemented within this architecture, *demon horde sort*, utilizes "hoarder" atoms that sort values spatially: the hoarders move about randomly in space, pushing locally higher values above itself, and locally lower values below itself [3]. Like the lymphocytes and ants, the hoarders act individually, only producing the desired end result of a sorted list when considered on a global scale. Since the behavior of each hoarder is essentially the same, a space filled with many hoarder atoms will contain a large amount of functional redundancy that allow

the system to tolerate many environmental perturbations, such as the destruction of hoarder atoms [2]. This can be thought of as a robust parallelized bubble sort in a sense, reaping the performance benefits of non-sequential computation while also being robust to faults [1, 9]. Ackley and Small's system is facilitated by the same concepts Mitchell identified in her work: decentralized systems harness the power of noise and redundancy to self-organize and perform complex tasks.

However, we feel that there is another potential source of variability that has not been considered as extensively in the work presented above: spatial irregularity. Both Messinger et al. and Ackley and Small draw inspiration from biological systems yet continue to utilize uniform grids as their spatial representations. Natural dynamical systems do not typically operate in a uniform lattice, so this may be a limiting assumption. Thus, we will present previous work considering spatial representations and how they may potentially impact the dynamics of the distributed CA systems we have been examining.

2.4 Spatial Representations and Modeling of CAs

2.4.1 Limitations of Traditional CA Spatial Representations

Returning to the applied CA models from section 2.1, there are some who question the viability of traditional CA models as adequate simulations of real-world phenomenon. Beyond the obvious limitation that real-world systems rarely exist in regular grids, a primary concern is that the grid predefines a fixed spatial scale, making representations of both cells and the environment itself inflexible [18]. In the case of modeling population dynamics, if the cell size is on the scale of the agent we are modeling, building and simulating on a grid that is adequately sized for modeling the movement of the agent often becomes intractable [6]. On the other hand, if the cell size is larger than the agents being modeled, the structure of the grid has placed an arbitrary bound on the population density it can represent.

Additionally, the neighborhood definitions are entirely dependent on the structure of the grid configuration, making it difficult to define alternative neighborhood stencils beyond variations of the standard Moore or Von Neumann neighborhoods. The local connectivity can be varied by using different shapes such as hexagons as tiles, but the number of neighbors per cells are still restricted to the same amount across the space. Thus, in order to support richer neighborhood interactions, the size of the neighborhood must be variable while still respecting the local connectivity of the grid. We consider a first step in this direction by examining alternative formulations of the Game of Life.

2.4.2 Spatial Variation of Conway's Game of Life

The Game of Life, discovered by John Conway, is a $K = 2$, $N = 9$ two-dimensional cellular automaton that is particularly remarkable because of both the rich emergent structures it can generate as well as its Turing universality [16]. Most instances of the Game of Life are played on regular, two dimensional lattices; what changes in behavior occur when it is simulated on an aperiodic *Penrose tiling*? Hill et al. examined this variation, and in particular ran experiments that explored the *lifetimes* (number of generations until stabilization) as well as *ash densities*

(the fraction of "on" cells in a stabilized configuration) of random initial configurations. The Game of Life on the Penrose tiles have lifetimes that are much shorter and ash accumulation half as dense as games played on the regular lattice [19]; this is likely due to the irregularity in the tiling distribution, illustrating that at least in this instance, the variable neighborhood sizes across the grid make it more difficult to maintain coherent structures. It should be noted however that Hill et al. directly translated the rules of the Game of Life onto the Penrose tiling without any modification, which may not be appropriate since a given cell may have either eight or nine neighbors as opposed to the constant eight neighbors in traditional Life instances. An alternative rule set would be to define the "living" criteria based on a ratio or percentage of live cells in the surrounding area, as it would produce appropriately dynamic behavior that aligns with the variability in the tilings. This addition is an extension on these initial experiments that may be more revealing to the overall relationship between the Game of Life played on a grid and more irregular environments like Penrose tiles.

To take spatial representations to one extreme, the concept of a grid can be removed altogether. *SmoothLife* is the result, where instead of cells on a grid individual points within the Cartesian space is considered and transition rules are defined for a point's circular neighborhood [34]. In SmoothLife's spatial representation, Euclidean distance is respected in regard to neighborhood definitions resulting in a true spatial representation of connectivity. Unfortunately, SmoothLife represents too radical a shift away from the discrete dynamic systems we have been considering with the lack of spatial structure entirely. Ideally, spatial representations of Euclidean distance and resulting neighborhood connections should be preserved while still maintaining a discrete structure. Voronoi diagrams possess these exactly traits, and so we will consider Voronoi-based CA systems next.

2.4.3 Voronoi Representations of CA

First, we review the basic idea of Voronoi diagrams. In order to construct a diagram, a set of *generator points* are placed in the plane. The cell defined by a generator point i is defined as the area containing all points of the plane that are closer to i than any other point with regards to Euclidean distance (Figure 2.11) [30]. In the realm of Geographic Information Systems, Voronoi CA systems have been utilized to explore irregular spatial patterns that occur in the real world [8, 36]. Shi and Pang suggest that the main barrier that prevents CAs from being applied in the real world is that CAs cannot provide a way to handle irregular, spatially defined neighborhood relations; Voronoi-based CAs appear to be a solution to this fundamental problem. Furthermore, due to their nice spatial properties, Voronoi diagrams are often utilized in models of various natural structures, such as cells [30]. Though there has been extensive work examining computation in regular dynamical systems (Section 2.2) and numerous applications of CAs both applied to and derived from nature (Section 2.1), there has been little work done analyzing how biologically plausible spatial representations such as Voronoi diagrams can impact computation and dynamics within a CA system. Flache and Hegselmann provide one such analysis in the context of modeling social dynamics.

Flache and Hegselmann simulate several social dynamics interactions through CA systems ran on both regular grids and Voronoi diagram analogs. The same global behavior was pre-

served in the Voronoi cases, though the dynamics took a longer time to converge to stabilized behavior relative to the regular grid case [15]. Though only a specific class of computational tasks were tested, the conclusion that these social dynamics tasks are robust to variation in grid structure leads us to the beginning stages of establishing some form of equivalence between classes of irregular and regular grid structures. Additionally, examining the spatial behavior of cells in the Voronoi case revealed some interesting patterns. Because of variations in the local structuring within the Voronoi diagrams, some areas of the irregular grid never participate in computation even when completely surrounded by computing cells, resisting all outside influences (Dead cells figure) [15]. This "dead cell" phenomenon provides evidence that the spatial properties inherent in Voronoi diagrams can impact the system's dynamics beyond variations in cell neighborhood connectivity. Additionally, the potential presence of dead cells along with the randomness produced by generating grids through generator points make Voronoi diagrams an interesting way of introducing variability to a cellular automata as it implicitly encodes some irregular features we have explored in Section 2.3. However, further work is needed to thoroughly investigate the impact of different spatial representations on the complex behavior and computation in dynamic, decentralized systems.

2.5 Summary

We have seen throughout this review that work on the computational dynamics of distributed natural systems is a two-way street, with researchers both utilizing cellular automata to investigate complex behavior in natural systems as well as drawing inspiration from nature to build more robust and efficient "cellular computers." Additionally, there has been extensive work concerning how dynamic systems can give rise to computation with an abundance of metrics that can be used to analyze such systems. It would be reasonable to apply the measures from these computational criticality studies to attempt to understand natural systems. Unfortunately, there is not a clear "mapping" between the foundational structures of cellular automata and dynamical natural systems. Assumptions cannot be made about a potential correspondence between local connections, transition rules, or spatial orientation when CAs and natural systems operate within such contrary environments: the control and precision of computer simulation versus the noise and inconsistency of the real world. For example, the spatial structure of cellular automata is limiting when utilizing them as a model (Section 2.4), which is why researchers in specific modeling domains have explored removing the lattice regularity of CAs in order to observe the impact that structural assumption has on the dynamic behavior of the system.

There is inherent imprecision when dealing with natural systems, which makes utilizing the criticality statistics problematic. The definitions of metrics like λ and γ as well as the notion of average dynamic behavior can be difficult translate to systems where there are no structural uniformities that can serve as a foundation for measurement. We have to adapt these metrics to the domain of irregular cellular automata in order to properly investigate the nature of computation in such systems. Perhaps there is some sort of functional equivalence between regular and irregular CAs, or perhaps they are separate classes of dynamical systems. Thus, this work will take some first steps towards resolving the complexion of this relationship. Our goal

is to examine and identify potentially fundamental differences or equivalences between irregular and traditional cellular automata in the pursuit of understanding how natural decentralized systems can compute "in the wild."

Figures

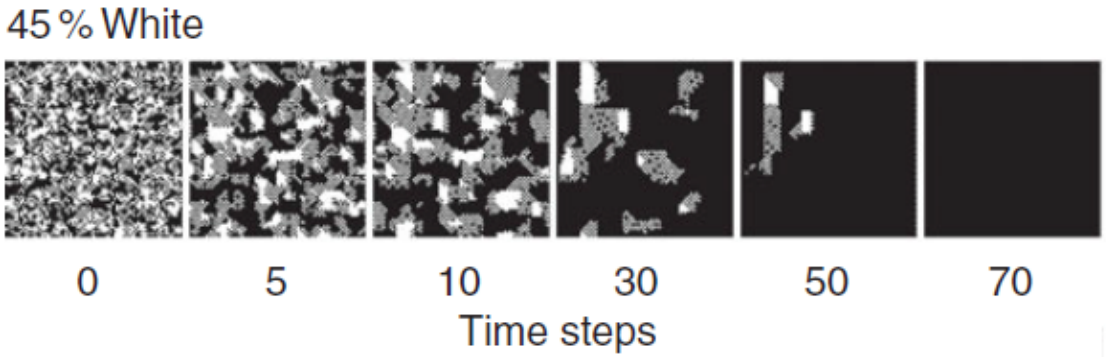

Figure 2.1: An illustration of a majority task cellular automata correctly classifying the majority initial state by converging to black. Figure from Mott and Peak [29].

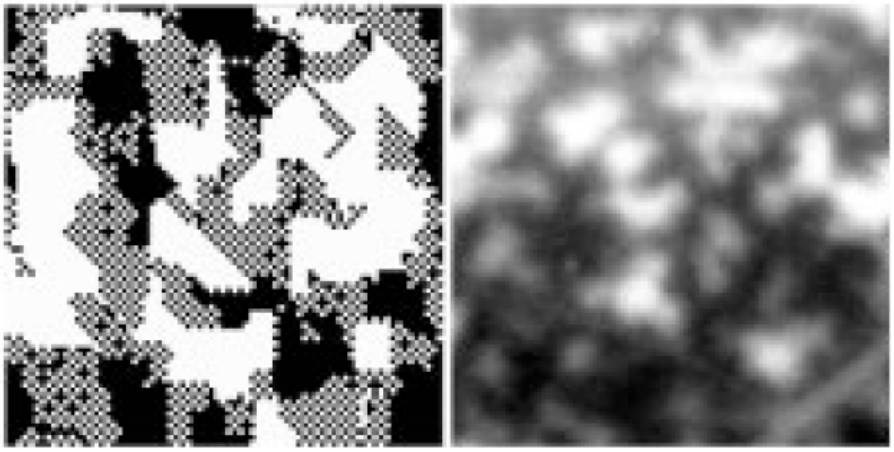

Figure 2.2: The left image illustrates coherent state patchiness in a majority task cellular automata. The right image illustrates a similar patchiness for stomata in the plant *Xanthium strumarium L.* Figure from Messinger et al. [25].

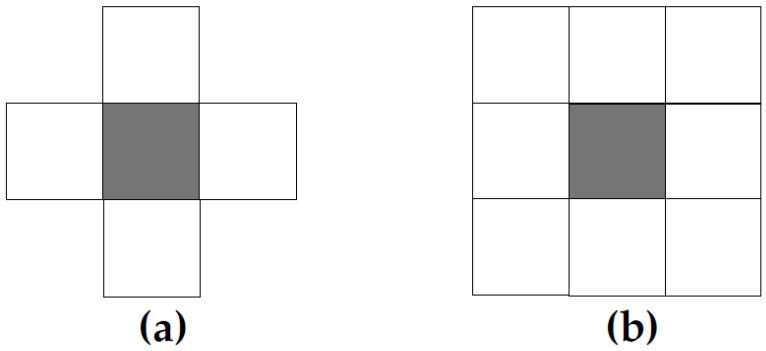

Figure 2.3: (a) is the five-cell von Neumann neighborhood, (b) is the nine-cell Moore neighborhood. The gray cell is the one to be updated by a transition rule. Figure from Mitchell et al. [28].

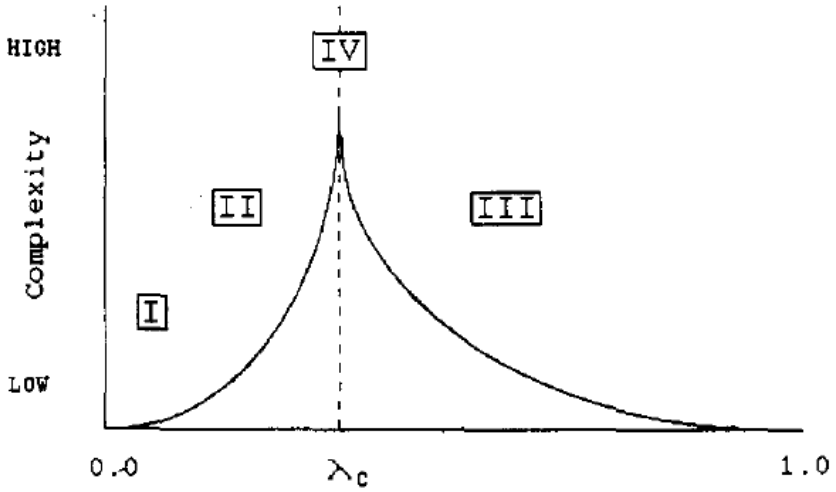

Figure 2.4: The hypothesized relationship between λ and complexity. Class IV CAs would only appear at critical levels of λ. Figure from Langton [21].

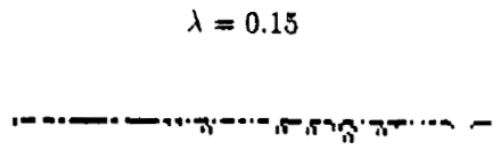

Figure 2.5: The progression of a one-dimensional $N = 4$, $K = 5$ cellular automaton with $\lambda = 0.15$ from a random initial configuration (time progresses from top to bottom). The transient length is incredibly short, with the automaton converging to a homogeneous state (all white) within four or five time steps. Figure from Langton [21].

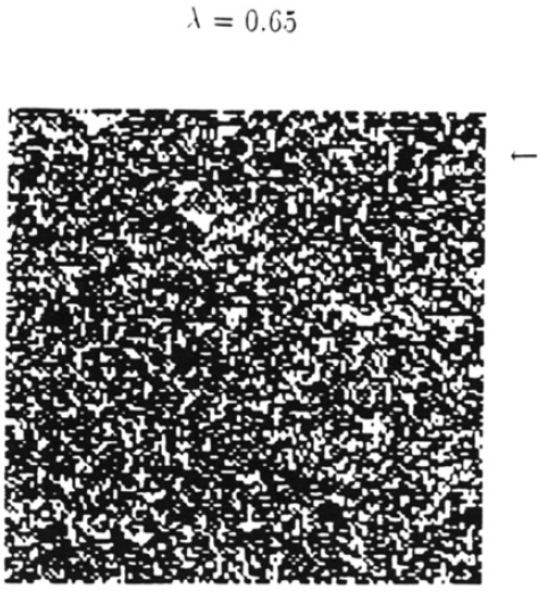

Figure 2.6: The progression of a one-dimensional $N = 4$, $K = 5$ cellular automaton with $\lambda = 0.65$ from a random initial configuration. The transient period ends quickly (indicated by the arrow), with dynamic activity degenerating into chaos. Figure from Langton [21].

Figure 2.7: The progression of a one-dimensional $N = 4$, $K = 5$ cellular automaton with $\lambda = 0.5$ from a random initial configuration. The transient length is quite long (over 10,000 time steps), indicative of more complex dynamical activity. Figure from Langton [21].

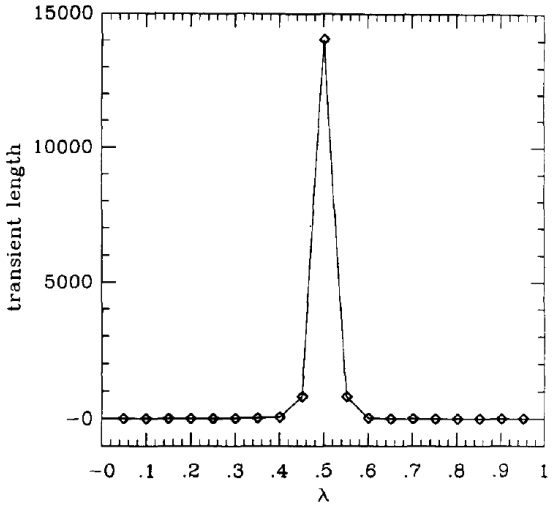

Figure 2.8: Average transient length as a function of λ for a 1D CA. The long transient lengths centered around $\lambda_c = 0.5$ indicate the transition between ordered and disordered dynamics. Figure from Langton [21].

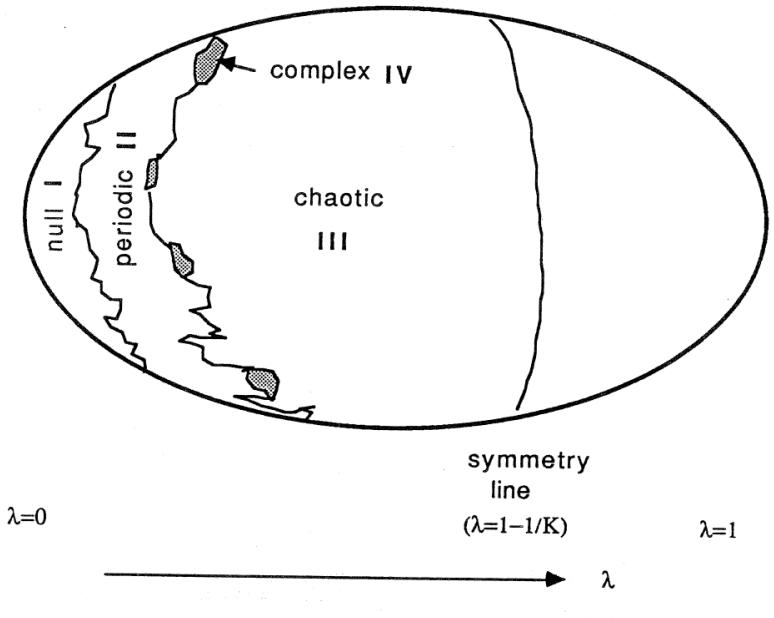

Figure 2.9: A hypothesized structure of the cellular automata rule space, with the relative location of Wolfram's complexity classes listed. In most areas, the transition between class II and class III automata is sharp. However, some subregions contain a smooth transition at the boundary, where class IV automata are hypothesized to reside in. Figure adapted from Li et al. [23].

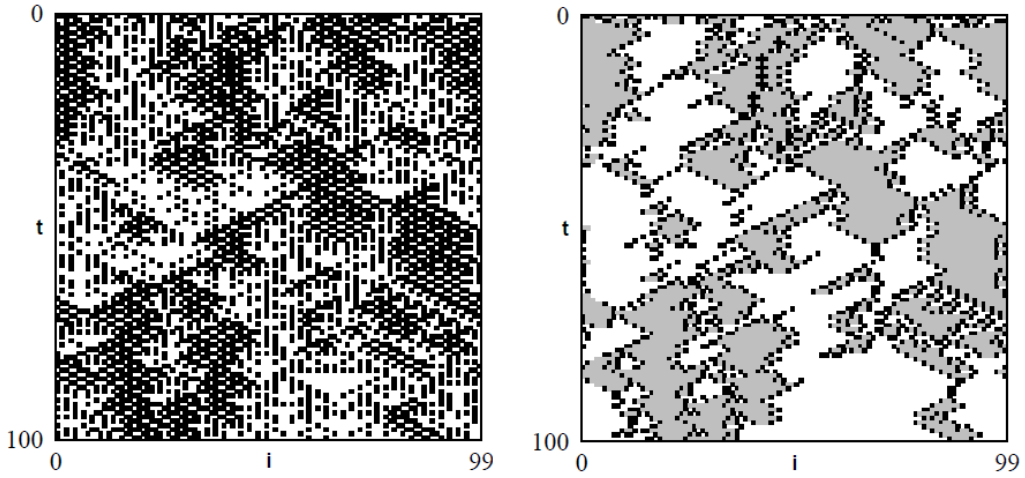

Figure 2.10: Space-time plots for a $K = 5$, $N = 2$ 1D cellular automata with a random initial configuration of lattice size 100. The left image illustrates the unfiltered progression of the CA, while the right image illustrates the same behavior, filtered by two regular domains indicated by white and gray. Figure adapted from Crutchfield and Hanson [10].

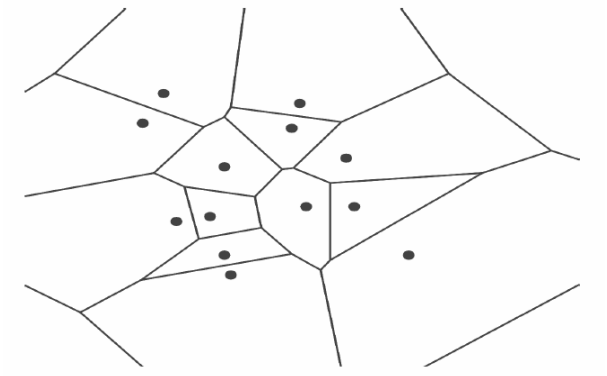

Figure 2.11: A voronoi diagram with its corresponding generator points. Figure adapted from Whigham and Dick [40].

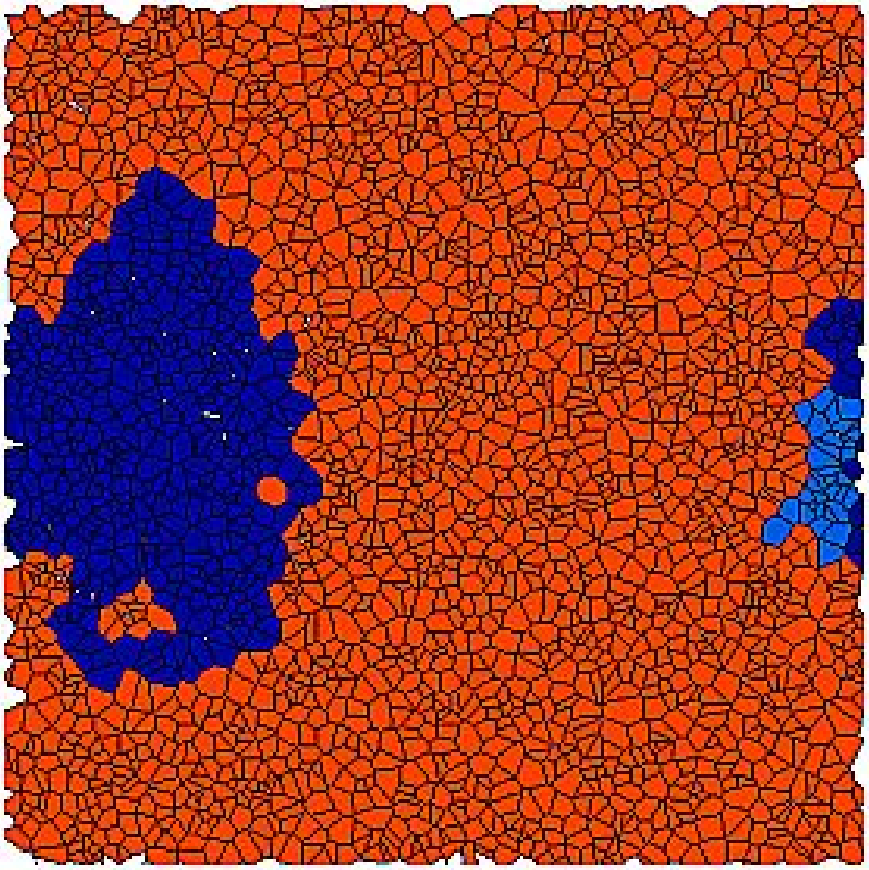

Figure 2.12: An irregular cellular automaton modeling cooperation dynamics in its stable state. Cooperating cells are blue and non-cooperating cells are red. Though the expected end state in this simulation on a regular grid would be a homogeneous cluster of cooperation, there are cells completely surrounded by cooperating neighbors that resist participating in the computation. Figure from Flache and Hegselmann [15].

Chapter 3

System Design

In this chapter, we will give an overview of the collection of tools we have built and utilized for our exploration of irregular CA systems. The goal is to provide enough information so that any experiments described in later chapters can be replicated.

3.1 System Overview

Our CA simulation platform follows an event-driven architecture written in C++; actions are placed on a queue and are performed one at a time. The event queue structure allows us to easily interleave actions in between time steps of a CA simulation; for example, we can take snapshots of the grid state or dump measurements to file in between time steps at any frequency we wish.

We designed this system primarily with flexibility in mind. Modular components and data structures can easily be swapped in and out of the simulation platform, giving us fine-grained control over the experiments we are running. Performance is not a major concern, so long as simulations and experiments can complete within a reasonable amount of time: for reference an experiment that runs 1,000 simulations 500 time steps each, writing data to disk at every time step for all simulations takes a few hours in total to complete.

Termination of the CA simulation occurs either when a maximum time step is reached, or when a grid state is repeated. We track grid states by walking through all cells on a grid in a consistent order, appending each cell state to a string array and then passing the resulting string through a hash function. We keep a history of these hashes along with the time step they occur, and terminate the simulation if a hash is ever repeated. Recording the time step also allows us to track the periodicity of any potential oscillating structures occurring within the simulation.

We execute rule table updates across the grid in a "spreading activation" fashion in order to increase the performance of the simulator. Instead of visiting and updating each cell, we track which cells changed state in the previous time step and place those cells along with their neighbors in a "change" list. When updating the graph for the next time step, we only apply the transition rule to cells in the list. We then repeat this process, updating the list with any cells and their neighbors that have changed state. We initialize the change list at the beginning of the simulation to all cells in the grid. This method of applying updates allows us to only consider

cells that could potentially change, saving time throughout the CA simulation especially if the grid is large.

3.1.1 Structures

Our CA simulation platform consists of a collection of data structures that are utilized in tandem to run the simulation. The most essential structures are briefly described below.

- **Grid Generators:** The `GridGenerator` structure stores both the geometric representation of a grid and the graph representation of a grid. The graph is implemented as an adjacency list as we expect the graph of a CA grid to be relatively sparse due to the strictly local connectivity.
- **Stencil:** A `Stencil` determines the local neighborhood for each cell in a Grid. Possible stencils include "generalized" von Neumman and Moore neighborhoods or "continuous" stencils that weight neighbors by their distance from the center cell.
- **Rule Table:** A `RuleTable` represents a transition rule, utilizing a Stencil to determine a cell's neighborhood and compute its next state.
- **Geometry Maps:** Maps and reverse maps from labels to geometric objects such as `Points`, `Edges`, and `Faces` are maintained for easy object access and grid manipulation.

Grid Generators will be reviewed in more detail in Section 3.2, while Stencils and Rule Tables will be discussed in Section 3.3. A schematic of our overall program architecture is pictured in Figure 3.1.

3.1.2 Input/Output Files and Running Our System

We have a standardized format for storing and reading in grid configurations, which allows us to easily pause and restore CA simulations as well as save and transfer them across different machines. A representative input grid file can be found in Appendix B.1. The visual representation of our grids are built by creating graph descriptions for **neato**, a graph layout program part of the Graphviz drawing package [14]. Images of our neato grid representations can be found in Section 3.2. Measured statistics are exported to .csv files and analyzed using Python with packages from the SciPy ecosystem [20].

We utilize the GNU C library package `getopt` to pass Unix-style command line arguments to our system, making experiments easily scripted and batched across multiple machines.

3.2 Grid Generation

In this section we'll describe the algorithms utilized to generate the various grids we run CA simulations on.

3.2.1 Penrose Tilings

To generate Penrose Tilings and Penrose-connected graphs, we use inflation processes to recursively generate the tiling. Robinson's triangle recursive decomposition technique is utilized to draw both kite/dart and thin/thick rhomb tilings, except full tiles are drawn instead of individual triangles [35]. Though drawing full tiles will likely result in tiles being considered and drawn multiple times, we avoid this by marking faces as they are drawn, only keeping a single copy of each face. As in all our grid generation programs, we track geometric primitives (`Point`, `Edge`, `Face`) to make generating our standard grid input file straightforward.

/* * This program generates images of penrose tilings and penrose-connected * graphs. Its main use is with Tony Liu's software to perform CA simulations * on general graphs. * * The approach used here is to recursively generate patches of a tiling * through inflation processes. It supports kite/dart and thin/thick rhomb * tilings by modeling Rapheal Robinson's recursive decomposition. However, * instead of drawing triangles, this program draws the full associated tile. * Thus many tiles encountered in the decomposition have the potential to * be drawn multiple times. To avoid this, we "'register" faces as they're * encountered, keeping only one copy of each encountered. * * Points that are mentioned during this process are stored in a point list. * The index of the point in this list is used, then, as a shorthand for * that location. Faces are collections of four point indices, with the * smallest index first, and the smallest neighbor second. For some * purposes (notably for the generation of Liu-style graphs), it is helpful * to have a notion of edges. When necessary, edges are constructed as pairs * of point indices with the smaller index first. * * All objects are placed in a coordinate space whose unit size is 1/72", * a PostScript point. This is arbitrary, but makes the generation of * PostScript output easier. To support affine transformations, a small * transformation system is provided that allows translation, rotation, and * scaling of 2D coordinate systems. It is compactly stored and models, loosely * the approach used by PostScript. For example, the transformation matrix * is stored as a 3x2 matrix since the last column can be fixed. When * necessary, the current graphics transform (currentMatrix) can be pushed * on a graphics context stack with gsave and later restored with grestore. * This allows routines to save and reset the graphics transformation system * for use in scaling, rotatating, and translating the final image in a * PostScript output (which centers the image in the center of a page) or * for supporting rescaling for use within a fixed coordinate viewport centered * around the origin. * * Registerface may fail to register the face (and thus not include it in * the final output) if all of the points are more distant than 'radius' away. * * All searches of lists are linear, and in many cases the searches are * obviously inefficient and could be improved with additional data structures. * * There is heavy use of dynamic memory allocation with little concern about * freeing or collecting garbage. Obviously, this could be improved. * * This software is free and distributed as-is, and is not suitable for any * particular purpose. */

3.2.2 Delaunay Triangulations and Voronoi Diagrams

The goal with building grids using Voronoi diagrams is to produce irregular grids that still respect Euclidean distance, as discussed in Section 2.4.3. Given a set of *generator points* $\{p_1, ..., p_n\}$ in the plane, a Voronoi polygon V_k corresponding to a generator point p_k consists

of all points whose distance to p_k is less than its distance to any other generator point. Thus, we can partition the plane into Voronoi polygons that correspond to every generator point, producing a *Voronoi Diagram*. These polygons will serve as the cells in our CA simulations.

The first step in the process to generate a Voronoi-based grid is to produce generator points. Though a simple uniform random distribution of points in the plane will generate a valid Voronoi Diagram, the resulting grid may have undesirable geometric properties due to the potential for points to clump together. Instead, we can utilize a method called *Poisson Disk Sampling* to generate random points that are guaranteed to be at least some specified distance apart from each other [7]. Points created by this method will necessarily avoid clumping, illustrated in Figure 3.2.

Once we have a set of generator points, we construct the dual graph of the Voronoi Diagram, known as the *Delaunay Triangulation*. The Delaunay Triangulation of a set of points $\{p_1, ..., p_n\}$ in the plane is a triangulation such that no point p_k resides inside the circumcircle of a triangle; an edge e is considered to be *locally Delaunay* if the two triangles that share e as a common edge satisfy the circumcircle condition [12]. We construct this dual graph first because (1) the edges of the Delaunay Triangulation exactly correspond to the graph connectivity of the Voronoi grid produced by the set of points and (2) the Voronoi Diagram is easy to construct from the triangulation. We utilize the *flip algorithm* to construct the Delaunay Triangulation: beginning with an arbitrary triangulation, edges that are not locally Delaunay are "flipped" such that they are locally Delaunay. Once there are no more edges to be flipped, then the resulting triangulation must be a Delaunay Triangulation [13]. Though the flip algorithm is not the fastest Delaunay Triangulation algorithm, it is straightforward, requires no auxiliary data structures, and performs well enough for our purposes.

Given a Delaunay Triangulation, we can produce its corresponding Voronoi Diagram by calculating the circumcenter of each Delaunay triangle and creating an edge between the circumcenters of adjacent triangles [12, 13]. The circumcenters of the triangles correspond to Voronoi polygon vertices, so we obtain valid Voronoi regions. Thus we have obtained both the graph representation and the geometric representation of a Voronoi-based grid. An illustration of this process is shown in Figure 3.3.

3.2.3 Voronoi Quadrilaterals

Experiments and simulations involving λ require static neighborhood sizes across all cells in the grid. One way we address this requirement is by generating irregular grids that have cells of the same number of sides. Specifically, given a Voronoi Diagram we can further partition the region such that all cells in the plane are quadrilaterals. This conversion is accomplished by taking two edge-adjacent Voronoi polygons and forming a quad from the two generator points and the end points of the shared common edge between the polygons (Figure 3.4) [32]. As long as the given Voronoi Diagram is "well-formed," specifically with all generator points placed at least some minimum radius from each other, this quadrilateral generation is possible across an entire Voronoi diagram, as seen in Figure 3.5. Note that though cells in the resulting *VQuad* grid always share edges with four other neighboring cells (excluding the boundary) resulting in uniform generalized von Neumman neighborhood sizes across the grid, the number of cells

they share vertices with are variable. This variability in vertex adjacency is partially due to the presence of concave quadrilaterals in the resulting grid.

3.2.4 Grid Degeneration

In Chapter 6, we consider various degenerated Voronoi Quad grids in our examination of the λ parameter. We will discuss the methods for generating these degraded grids here.

Generator Point Removal

As noted in Section 3.2.3, VQuad grid generation is dependent on the position of Voronoi generator points in the input Voronoi diagram. Thus, one manner of grid degeneration is simply to remove generator points from the input diagram. Since a generator point in the input diagram could be a vertex to many VQuad cells, generator point removal has the effect of taking out large chunks of the VQuad grid, as pictured in 3.6.

Crosshatching Degeneration

Though generator point removal provides a simple way to degrade a grid, we wanted finer-grained control over how points and edges are removed. Thus we devised a manner of degradation that allowed us to specify what local regions of the grid would be disconnected from other portions of the grid. This basic premise is to draw vertical and horizontal lines at regular width w intervals apart that subdivide the irregular grid roughly into square lattice regions. Cell edges that intersect these *crosshatchings* are removed from the grid with some probability p: cells that were formerly connected by the edge no longer are neighbors in the graph representation of the grid. Thus at $p = 1.0$ we would have completely isolated "islands" of grid subregions. This *crosshatching degeneration* technique allows us control over the degree of regional connectivity, as pictured in Figure 3.7. This manner of degradation preserves the number of cells present in the grid, instead introducing degradation by decreasing edge adjacency. Though we primarily use crosshatching degeneration as a form of edge removal, this technique also gives us a way to control generator point removal: since the crosshatchings define a square lattice partitioning of the grid, we can also choose square regions to remove with some probability p. Any generator points sitting within the boundary of a chosen square region will be removed from the graph. Thus we can control both the frequency of generator points removed (through adjusting p) as well as the average size of the of the regions removed (through adjusting w). This crosshatching technique allows us to parameterize the degeneration of a grid beyond simply measuring average neighborhood size.

3.3 Stencil and Rule Table Implementation

3.3.1 Generalized von Neumman and Moore Neighborhoods

We would like to maintain a notion of both von Neumman and Moore neighborhoods even on irregular grids. Thus we formally define *generalized* notions of both neighborhood stencils: a *generalized von Neumman neighborhood* for a cell c is defined as all cells that share an edge with

c, while a *generalized Moore neighborhood* for c is defined as all cells that share a vertex with c. These generalized neighborhood definitions are not necessarily equivalent to their standard definitions. For example, while the generalized von Neumman neighborhood for Voronoi Quad grids is identical to the regular case, the generalized Moore neighborhoods are not uniform across the grid and may have neighborhood sizes ranging from 6 to 12. Because we maintain geometry maps in our system, defining either neighborhood stencil is straightforward for any CA simulation on any grid.

3.3.2 λ Rule Tables

In the λ experiments performed by Wootters and Langton, rule tables must be *isotropic* in that all planar rotations of a particular neighborhood configuration map to the same output cell state; this condition removes the global property of spatial orientation from influencing rule transitions [4, 43]. As a result, what would be a K^N sized table results in a smaller table because only rotationally distinct neighborhood configurations are unique entries: specifically, if we consider the $K = 8$, $N = 5$ case the rule table of size $8^5 = 32,768$ is reduced to a rule table of size $8,352$ (see Appendix B.2 for calculation). To generate these tables, we iterate over all K^N possible neighborhood values, canonicalize their string representations by finding their *lexicographically minimal rotation*, and track only the unique neighborhoods. For example, in the $K = 8, N = 5$ case, neighborhoods "0110" and "1001" map to the same minimal rotation neighborhood of "0011". We then append a center state value to the end of this canonical neighborhood representation, thus creating the keys for our λ rule table mapping with the output state for that particular neighborhood configuration as the value.

3.4 Summary?

Figures

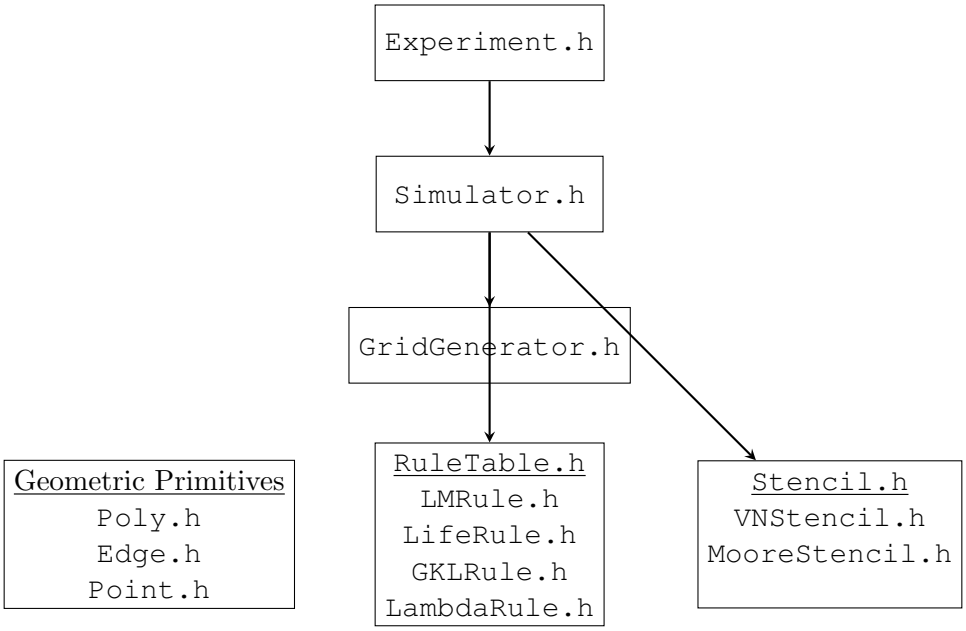

Figure 3.1: CA Simulation Class Architecture

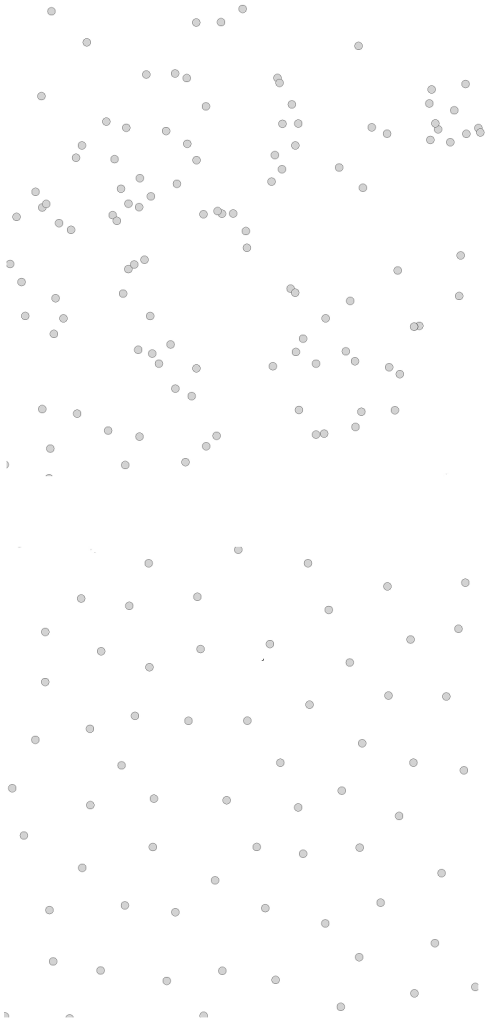

Figure 3.2: On top, a random uniform distribution of points. On bottom, a random distribution of points generated by Poisson Disk Sampling.

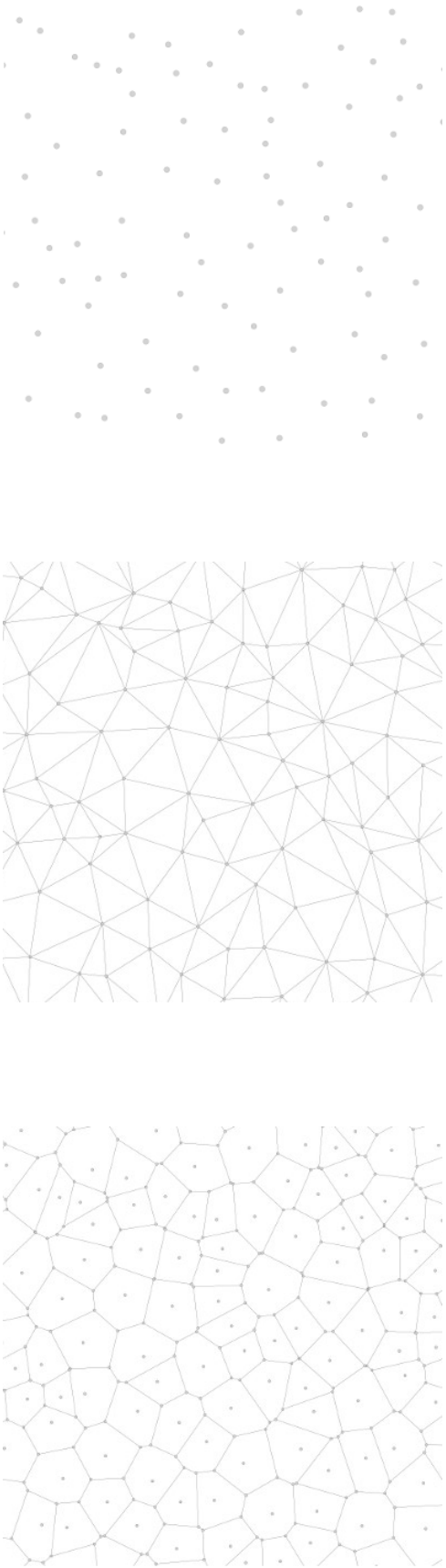

Figure 3.3: The Voronoi Diagram construction process. We begin with a random set of points (top), compute the Delaunay Triangulation (center), and subsequently the Voronoi Diagram (bottom).

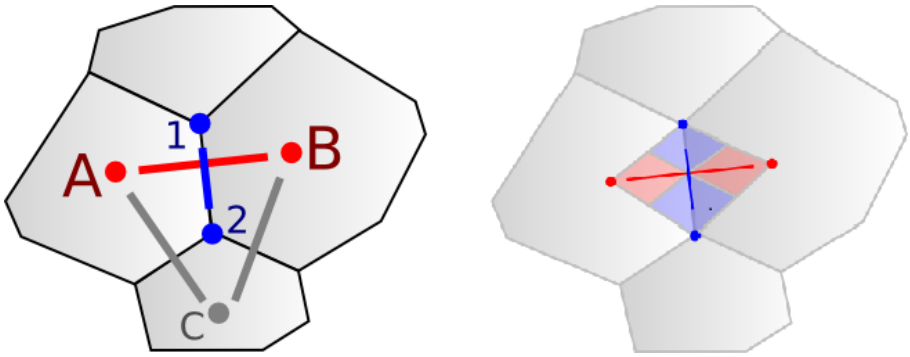

Figure 3.4: An illustration of the Voronoi quadrilateral creation process. Generator points A and B are connected with the shared edge endpoints 1 and 2 to create the quadrilateral pictured on the right. Figure adapted from Patel [32].

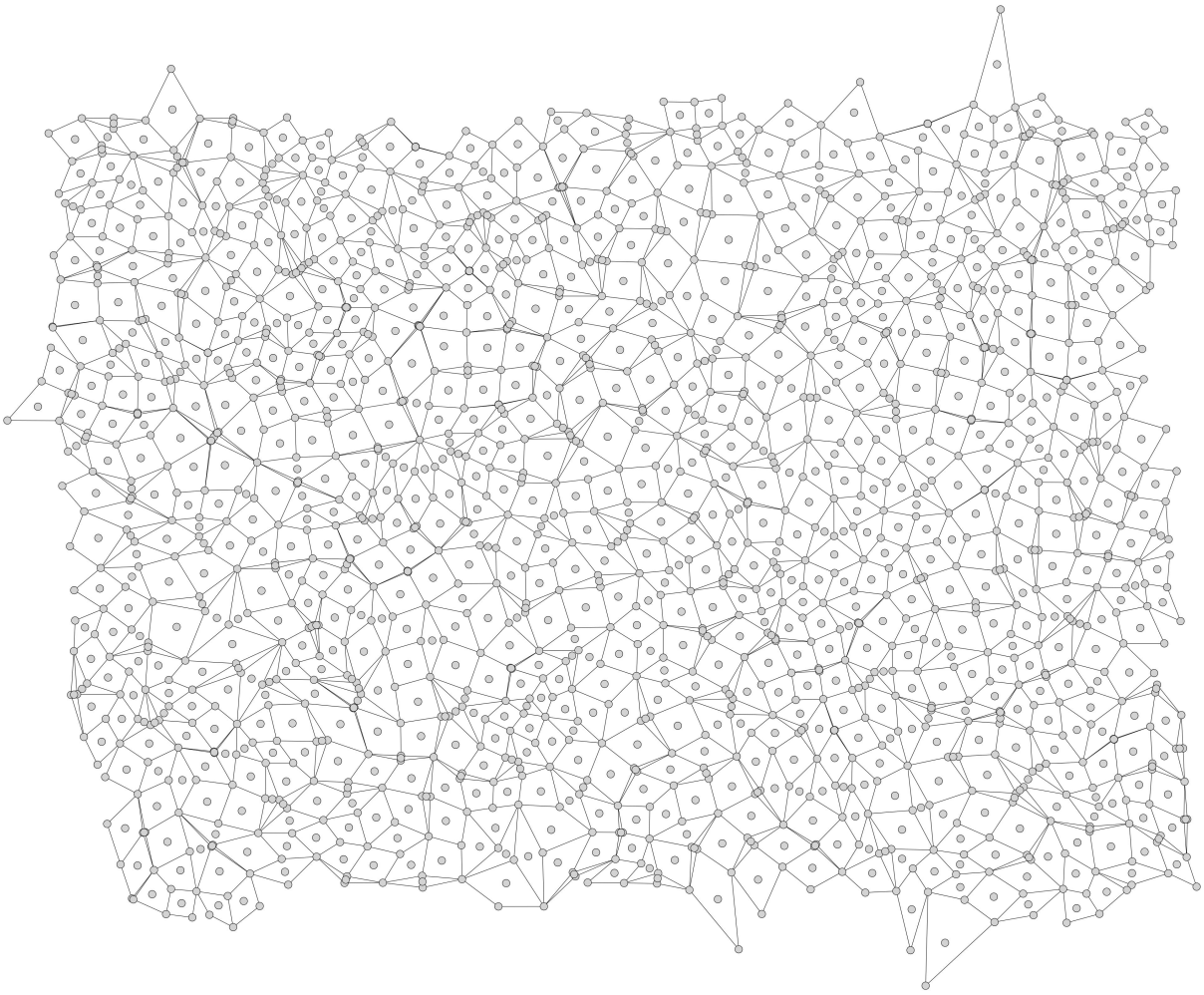

Figure 3.5: A representative Voronoi Quad grid, generated from the same set of generator points as Figure 3.3.

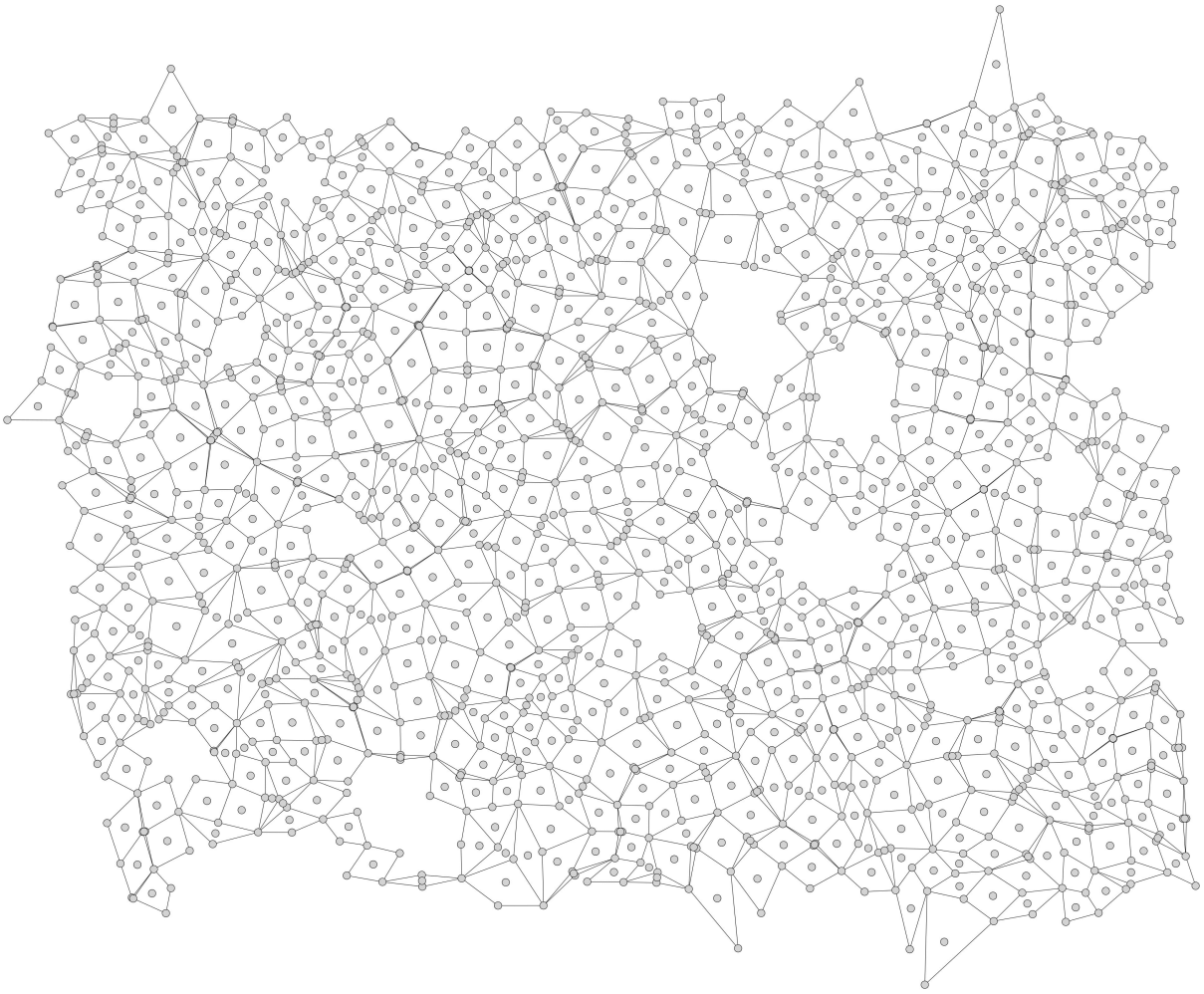

Figure 3.6: A VQuad grid degraded by the removal of 7% of the input generator points. Notice the lack of small, (particularly single cell) "holes" in the resulting grid.

Figure 3.7: Crosshatching Degeneration caption TODO

Chapter 4

Penrose Life Experiments

We begin our exploration of irregular CA by considering aperiodic tilings. Specifically, we build upon work done by Hill et al. (see Chapter 2.4.2) in examining the Game of Life on Penrose Tilings. Aperiodicity is a move away from the regularity found in square lattices but still maintains some local neighborhood structure.

4.1 Replication and Extension of Hill et al.'s Work

-

TODOs

- run lambda experiments for K=3, N=11 grids? (penrose glider)

Chapter 5

Local Majority Experiments

Chapter 6

Lambda Degradation Experiments

Chapter 7

Future Work

- majority task experiments beyond local majority
- adaptation of γ parameter to irregular 2D grids

Chapter 8

Conclusion

Appendix A

Definitions of Criticality Metrics

A.1 Lambda

(From Langton [21]) Given a cellular automata with cell states Σ ($K = |\Sigma|$) and N neighbors, pick an arbitrary state $s \in \Sigma$ and call it the *quiescent* state s_q. Let n be the number of transitions some transition function Δ maps to s_q. Pick the remaining $K^N - n$ transitions to be distributed uniformly over the other $K - 1$ states. Then λ is:

$$\lambda = \frac{K^N - n}{K^N} \tag{A.1.1}$$

A.2 Gamma

(From Li et al. [23]) First, we define the *left-moving difference rate* for one-dimensional CA by taking a configuration $a = \{...a_{-1}, a_0, a_1...\}$ and constructing another configuration $a' = \{...a'_{-1}, a'_0, a'_1...\}$ with $a_i = a'_i$ for $i < 0$, $a_i \neq a'_i$ for $i = 0$ and a'_i randomly chosen for $i > 0$. The *difference pattern* at time step t for a given cell i is defined as $\delta_i^t = 0$ if $a_i^t = a_i'^t$ and $\delta_i^t = 1$ otherwise. The *left front* of the difference pattern is $i_{left}^t = \min\{i|\delta_i^t = 1\}$. Thus the left-moving difference rate γ_{left} is:

$$\gamma_{left}(a) = \lim_{t,\tau\to\infty, t/\tau\to 0} = \frac{i_{left}^{\tau} - i_{left}^{\tau+t}}{t - \tau} \tag{A.2.1}$$

Similarly, the *right-moving difference rate* is defined with $a_i = a'_i$ for $i > 0$, $a_i \neq a'_i$ for $i = 0$ and a'_i randomly chosen for $i < 0$. The *right front* is $i_{right} = \max\{i|\delta_i^t = 1\}$. Then γ_{right} is:

$$\gamma_{right}(a) = \lim_{t,\tau\to\infty, t/\tau\to 0} = \frac{i_{right}^{\tau+t} - i_{right}^{\tau}}{t - \tau} \tag{A.2.2}$$

The overall difference spreading rate γ is then:

$$\gamma(a) = \gamma_{left}(a) + \gamma_{right}(a) \tag{A.2.3}$$

A.3 Shannon Entropy

(From Li et al. [23]) Given some discrete probability distribution $\{p_i\}$, the *Shannon entropy* H is defined as:

$$H = \sum_i p_i \log(p_i) \tag{A.3.1}$$

The *single cell spatial entropy* is calculated by defining a probability distribution counting the frequency of occurrence of all states in Σ for a set amount of time steps. Let $\{c_i\}$ for $i = 1...K$ be the set of all counts, where c_i is the number of times state $i \in \Sigma$ occurs. Then the probability distribution is:

$$\{p_i\} = \left\{ \frac{c_i}{\sum_j c_j} \right\} \tag{A.3.2}$$

A.4 Mutual Information

(From Li et al. [23]) Given two probability distributions $\{p_i\}$ and $\{p_j\}$ as well as their joint distribution $\{p_{ij}\}$, we define the *mutual information* M as:

$$M = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j} \tag{A.4.1}$$

We can apply mutual information to spatial-temporal patterns by defining the two probability distributions as the frequency of having a particular cell state on two sites separated by some distance d. The mutual information between two sites of any spatial-temporal distance can be computed.

A.5 Mean Field Theory Estimates

(From Li et al. [23]) Given a 1D cellular automata with $K = 2$ and $N = 2r + 1$, we can obtain some estimates of both λ_c and γ. λ_c can be estimated by:

$$\lambda_c = \frac{1}{2} - \frac{1}{2}\sqrt{1 - \frac{2}{r+1}} \tag{A.5.1}$$

In order to estimate γ, first calculate λ using the cellular automata's transition rules Δ. Then:

$$\gamma = \frac{(2r+2)\lambda - (2r+2)\lambda^2 - 1}{\lambda(1-\lambda)} \tag{A.5.2}$$

Wootters and Langton utilize mean-field theory to estimate the entropy H of a 2D cellular automata with $K = 8$ and $N = 5$ [43]. Let v be the density of the quiescent state s_q in the spatial configuration, so $v = 1$ would indicate a configuration of all s_q. $K = 8$, $N = 5$ Cellular automata in their steady state will satisfy the following equation:

$$v = v^5 + (1 - v^5)(1 - \lambda) \tag{A.5.3}$$

Then, assuming that the other seven states appear equally often in the transition rules Δ, we can estimate H:

$$H = -\left[v \log v + 7\left(\frac{1-v}{7}\right)\log\left(\frac{1-v}{7}\right)\right] \tag{A.5.4}$$

Appendix B

Miscellaneous

B.1 Grid Input File Format

B.2 Calculating Unique Rotations in a λ Transition Table

TODO

Figures

Figure B.1: TODO

Bibliography

[1] David H Ackley. Beyond efficiency. *Communications of the ACM*, 56(10):38–40, 2013.

[2] David H Ackley and Trent R Small. Indefinitely scalable computing= artificial life engineering. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 3–5.

[3] David H Ackley, Daniel C Cannon, and Lance R Williams. A movable architecture for robust spatial computing. *The Computer Journal*, page bxs129, 2012.

[4] Jeremy Avnet. Computation, dynamics and the phase-transition, 2000.

[5] Per Bak, Chao Tang, and Kurt Wiesenfeld. Self-organized criticality. *Physical review A*, 38(1):364, 1988.

[6] M Bithell and WD Macmillan. Escape from the cell: spatially explicit modelling with and without grids. *Ecological Modelling*, 200(1):59–78, 2007.

[7] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH*, volume 2007, page 5, 2007.

[8] Joao P Carvalho, Marco Carola, and José AB Tomé. Using rule-based fuzzy cognitive maps to model dynamic cell behavior in voronoi based cellular automata. In *2006 IEEE International Conference on Fuzzy Systems*.

[9] K Mani Chandy. *Parallel program design*. Springer, 1989.

[10] James P Crutchfield and James E Hanson. Turbulent pattern bases for cellular automata. *Physica D: Nonlinear Phenomena*, 69(3):279–301, 1993.

[11] James P Crutchfield and Melanie Mitchell. The evolution of emergent computation. *Proceedings of the National Academy of Sciences*, 92(23):10742–10746, 1995.

[12] Satyan L Devadoss and Joseph O'Rourke. *Discrete and computational geometry*. Princeton University Press, 2011.

[13] Herbert Edelsbrunner and Zhiqiang Gu. *Design and Analysis of Algorithms*. Duke University, 2008.

[14] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphvizopen source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2001.

[15] Andreas Flache and Rainer Hegselmann. Do irregular grids make a difference? relaxing the spatial regularity assumption in cellular models of social dynamics. *Journal of Artificial Societies and Social Simulation*, 4(4), 2001.

[16] Martin Gardner. Mathematical games: The fantastic combinations of john conways new solitaire game life. *Scientific American*, 223(4):120–123, 1970.

[17] David Griffin. Decoding the consensus problem. Personal communication, 2015.

[18] Rainer Hegselmann and Andreas Flache. Understanding complex social dynamics: A plea for cellular automata based modelling. *Journal of Artificial Societies and Social Simulation*, 1(3):1, 1998.

[19] Margaret Hill, Susan Stepney, and Francis Wan. Penrose life: ash and oscillators. In *Advances in Artificial Life*, pages 471–480. Springer, 2005.

[20] Eric Jones, Travis Oliphant, and Pearu Peterson. {SciPy}: open source scientific tools for {Python}. 2014.

[21] Chris G Langton. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1):12–37, 1990.

[22] Wentian Li and Norman Packard. The structure of the elementary cellular automata rule space. *Complex Systems*, 4(3):281–297, 1990.

[23] Wentian Li, Norman H Packard, and Chris G Langton. Transition phenomena in cellular automata rule space. *Physica D: Nonlinear Phenomena*, 45(1):77–94, 1990.

[24] Norman Margolus. Cam-8: a computer architecture based on cellular automata. *Pattern Formation and Lattice-Gas Automata*, pages 167–187, 1996.

[25] Susanna M Messinger, Keith A Mott, and David Peak. Task-performing dynamics in irregular, biomimetic networks. *Complexity*, 12(6):14–21, 2007.

[26] Melanie Mitchell. Self-awareness and control in decentralized systems. In *AAAI Spring Symposium: Metacognition in Computation*, pages 80–85, 2005.

[27] Melanie Mitchell, Peter T Hraber, and James P Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7(Santa Fe Institute Working Paper 93-03-014):89–130, 1993.

[28] Melanie Mitchell et al. Computation in cellular automata: A selected review. *Nonstandard Computation*, pages 95–140, 1996.

[29] Keith A Mott and David Peak. Stomatal patchiness and task-performing networks. *Annals of botany*, 99(2):219–226, 2007.

[30] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009.

[31] Norman H Packard. *Adaptation toward the edge of chaos*. University of Illinois at Urbana-Champaign, Center for Complex Systems Research, 1988.

[32] Amit Patel. Polygonal map generation for games. `http://www-cs-students.stanford.edu/ amitp/game-programming/polygon-map-gener` 2010.

[33] David Peak, Jevin D West, Susanna M Messinger, and Keith A Mott. Evidence for complex, collective dynamics and emergent, distributed computation in plants. *Proceedings of the National Academy of Sciences of the United States of America*, 101(4):918–922, 2004.

[34] Stephan Rafler. Generalization of conway's" game of life" to a continuous domain-smoothlife. *arXiv preprint arXiv:1111.1567*, 2011.

[35] Raphael M Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones mathematicae*, 12(3):177–209, 1971.

[36] Wenzhong Shi and Matthew Yick Cheung Pang. Development of voronoi-based cellular automata-an integrated dynamic model for geographical information systems. *International Journal of Geographical Information Science*, 14(5):455–474, 2000.

[37] Moshe Sipper. *Evolution of parallel cellular machines*. Springer Heidelberg, 1997.

[38] Moshe Sipper. The emergence of cellular computing. *Computer*, 32(7):18–26, 1999.

[39] Jevin D West, David Peak, Keith Mott, and Susanna Messinger. Comparing the dynamics of stomatal networks to the problem-solving dynamics of cellular computers. In *Unifying Themes in Complex Systems*, pages 327–341. Springer, 2011.

[40] Peter A Whigham and Grant Dick. A voronoi-based distributed genetic algorithm. 2003.

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

[42] Stephen Wolfram et al. *Theory and applications of cellular automata*, volume 1. World Scientific Singapore, 1986.

[43] William K Wootters and Chris G Langton. Is there a sharp phase transition for deterministic cellular automata? *Physica D: Nonlinear Phenomena*, 45(1):95–104, 1990.