Memory in Media with Manufacturing Faults

Mark A. McCann

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
By the Department of
Computer Science

September 2007

Abstract

As computational systems scale to *arbitrarily large* sizes, and as they are expected to function reliably for *arbitrary lengths* of time, there are certain realities of the physical world that can not be ignored in the design and modeling of computational systems. For example, considerations for the delivery of power and removal of heat seem to limit the system to a two-dimensional surface. Additionally, it can be argued that all of the following restrictions are desirable and reasonable: there is a finite variety of components, no components (or external agents) are immune to failure, and components can communicate only with a bounded number of other components over a bounded distance. A very general model, with a reasonable ability to capture many of these features, is that of a cellular automaton.

We extend the widely studied model of *transient faults* (which occur independently at different places and different times) in cellular automata to consider *manufacturing faults* (which occur independently at different places, but affect cells for all time). Although a well known monotone binary transition rule (known as Toom's Rule) in two dimensions can remember a bit (that is, the system can be used to preserve a single Boolean value for all time with probability one) in the presence of transient faults, we show that no monotone binary transition rule in two dimensions can remember a bit when both manufacturing faults and transient faults are present. On the other hand, we give a monotone binary transition rule in three dimensions that *can* remember a bit in the presence of both manufacturing faults and transient faults. (By adding one or two further dimensions, one can reduce the problem of performing an arbitrary computation reliably to the problem of remembering a single bit.) We also study cellular automata that are based on hyperbolic (rather than Euclidean) tessellations (including infinite regular trees), and we completely classify the cases in which majority voting among all nearest neighbors can tolerate manufacturing faults and/or transient faults.

Acknowledgments

First and foremost, I would like to thank my advisor, Nicholas Pippenger. I especially thank him for his patience, guidance, advice, and his constant encouragement (especially when I really needed encouragement!). It has been a great privilege and joy to work with Nick and to learn from him.

Just as importantly, I want to thank my wonderful family. My wife, Karen, has steadfastly supported me in all aspects of day-to-day life, from things mundane to life's biggest challenges. I have particularly appreciated her presence as a calming and organizing force during my time at Princeton. She is truly my best friend. My parents have been there for me from the beginning. I have always had their support, their encouragement, and their love throughout my many years in school. My sister, Erin, has been a wonderful listener and counselor, and I greatly value our friendship. I would not be where I am without them.

Mark A. McCann
Princeton University
June 2007

I gratefully acknowledge funding through Nicholas Pippenger's National Science Foundation grant CCF 0430656.

Contents

Chapter 1

Introduction

It was in 1952 that von Neumann [32] first broached the problem of performing arbitrarily large computations reliably when the basic computational operations are unreliable. He considered circuits in which each gate failed independently with probability $\varepsilon > 0$. He showed that for every $\delta > 0$ there exists an $\varepsilon > 0$ such that, for every circuit C built from perfectly reliable gates, there exists a circuit C' built from ε-failing gates that computes the same function as C except for a probability of error at most δ. One problem with von Neumann's construction (and all subsequent circuit-based work) is that it calls for arbitrarily long wires, and it requires the assumption that the failure probability for these wires is independent of their length.

One model that resolves this objection is that of cellular automata, as described by Ulam [31] and von Neumann [33]. Cellular automata naturally model the assumptions we would like to make about arbitrarily large computational systems in the physical world: each cell has finitely many states and communicates with finitely many neighbors at a bounded[1] distance from it. Furthermore, the system is completely homogeneous: the behavior of a cell does not depend on its location or on time. These assumptions give a scale-independent computational model in which the assumptions that individual components must satisfy are independent of the size of the computation. (A quite different notion of fault-tolerant computation is exemplified by the work of Kaklamanis *et al.* [18], who consider arrays of general-purpose processors which communicate by message-passing, which fail only by stopping, and which are reconfigured externally to route messages around externally diagnosed faults.)

There are many ways to define a cellular automaton, but all descriptions differ essentially only in detail or in the degree of generality. We give a more general definition of cellular automata in Section 1.1, but for now we give the following definition. There is an infinite collection of cells called the *cellular space* which are located at the points of a simple lattice $\mathbf{Z}^d$ for some $d \geq 1$. At every point of

[1] When we write "bounded," we generally mean that the same bound holds for all objects in question.

discrete time $t \in \{0, 1, \ldots, \}$, each cell is in a state taken from $\mathbf{B} = \{0, 1\}$. There is a common transition rule denoted $\phi = (x_1, \ldots, x_n, f)$, where f is a Boolean function of n arguments and $x_1, \ldots, x_n$ is a list of n offset vectors from $\mathbf{Z}^d$. The state of any cell $v \in \mathbf{Z}^d$ at time $t+1$ is determined by Boolean function f evaluated on the states of the cell's n neighbor cells at time t: $v + x_1, v + x_2, \ldots, v + x_n$. Thus, if the state of the automaton is specified at time $t = 0$, then the state of the automaton is fully determined at all subsequent steps of time and we call this sequence of states a *trajectory*. All updates to cells occur simultaneously.

The model of incorrectness for cellular automata most frequently considered by researchers up until now is as follows. Given a deterministic cellular automaton (as described above) in a known state at time $t = 0$, independently at each successive timestep and independently for each cell, with probability $\alpha > 0$, an all-knowing adversary (who knows everything about the past and future) is given the opportunity to set the current state of the cell. We call this the *transient fault model* because the behavior of a cell is only violated during the timestep when a fault occurs. On the next timestep, the cell follows its deterministic rule unless another failure occurs. We say that a (deterministic) automaton can *tolerate transient faults* for a particular trajectory if given any $\delta > 0$, there exists $\alpha > 0$ such that at any given time the probability of a given cell being in *error* (that is, being in a state differing from that of the deterministic trajectory) is at most δ.

The first published work on faults in cellular automata appears to be that of Stavskaya and Piatetski-Shapiro [27] in 1968. There is an important distinction between the fault model in [27] and the one we just described. In [27], faults occur "reliably" with probability α. There is no adversary choosing the states of at-fault cells, and the model is best viewed as an infinite Markov process. An objection to this model is that "reliable" faults provide a source of random coin-flips and it is quite possible that such an automaton might cease to work correctly in the absence of faults. This is one reason why we choose to let an adversary pick the outcome of random faults. Even when the Markov-process fault model is chosen, positive results are usually proven using the adversarial model. Of course, the Markov-process fault model yields stronger negative results, but very few such negative results are known even with fairly strong restrictions on the transition rules (see however [23] and our Theorem 30). For most of this thesis, we assume the use of fault models with an adversary.

An important advance in the study of automata with transient faults was the description of a simple rule by Toom [28], which we call "Toom's Rule," that is able to remember the "all zeros" and "all ones" trajectories in the presence of transient faults. The cellular space for Toom's Rule is the grid $\mathbf{Z}^2$. At each timestep, each cell takes a majority vote of itself and the immediate northern and eastern cells. The leftmost diagram in Figure 1.1 on page 10 illustrates Toom's Rule. It is easy to see that such a rule can dissolve any finite island of errors in a finite amount of time in the absence of further faults. If one draws an n by n square about the entire region of error, then at each subsequent timestep a

new diagonal line of corrected cells advances starting from the northeast corner and sweeping towards the southwest corner. After $2n$ timesteps, the entire island disappears. In [28], Toom proved that this simple rule is tolerant of transient faults.

In [14], Gács and Reif described how to perform a transient-fault-tolerant computation in three dimensions using a version of Toom's Rule. Roughly, the idea is to simulate a fault-free one-dimensional automaton[2] using a stack of two-dimensional slices[3] to remember the state of a single cell of the one-dimensional automaton. The third dimension is then used to perform computation. A version of Toom's Rule able to remember a finite number of states is used for each slice. However, even if each slice is able to remember only a single bit, we can implement Conway's "Game of Life" [2] (which is Turing complete) where each two-dimensional slice is used to remember a single bit and two additional dimensions implement the game itself. Thus, the problem of remembering a single bit can be seen as fundamental to computation. The majority of the results in this thesis focus on the problem of remembering a single bit.

A major assumption in the transient fault model is that faults for each cell only last for one moment of time. However, it seems implausible that an arbitrarily large cellular automaton could be totally free from permanent defects. We term such defects as "permanent faults" because an affected cell malfunctions (at the whim of the adversary) at every future point in time. If permanent faults are permitted to accumulate then they will eventually overwhelm any system. Thus, some external agent is needed to repair such faults as they occur. This would seem to require a significantly different type of model than simple cellular automata provide and we do not consider such a model in this thesis. Instead, we introduce a limited type of permanent fault that we call a *manufacturing fault*.

In the manufacturing fault model, we assume that independently, with probability β, a manufacturing fault occurs at each cell. If a manufacturing fault occurs, then the adversary is in control of the state of that cell at all times. We will generally consider transient faults in combination with manufacturing faults in what we call the *combined fault model*. We say that a cellular automaton is *tolerant of combined faults* with respect to a particular trajectory if for every $\delta > 0$, there exists $\alpha > 0$ and $\beta > 0$ such that the probability at every subsequent timestep that a given cell is in error is at most δ.

Immediately, we see that combined faults are fatal to Toom's Rule. Consider any cell and call it the origin. There is certainly a cell with a manufacturing fault somewhere directly to the east. Assume that all cells are initialized to a common state, say zero, and that the adversary leaves any cell with manufacturing fault permanently in the opposite state, say one. After sufficiently many timesteps, the cell immediately to the west will experience a transient fault and the adversary

[2] A one dimensional cellular automaton can easily implement a Turing machine as an iterative array.

[3] Each "slice" is a grid of cells on the $\mathbf{Z}^2$ lattice.

will set the state of that cell to one. Due to the manufacturing fault in its eastern neighbor, the cell now remains in error forever. Thus, eventually a stable chain of errors will reach back to the origin cell, all supported by a single manufacturing fault.

Given the failure of Toom's Rule to cope with manufacturing faults, a first question might be to ask if there are any cellular automata able to remember a bit on $\mathbf{Z}^2$ with manufacturing faults. For a fairly large and natural class of cellular automata, we show that the answer is negative. This negative result, along with the desire to study cellular spaces embedded in two-dimensional surfaces, inspires us to investigate the problem of memory on automata on non-Euclidean two-dimensional surfaces. The following is a brief overview of each chapter in this thesis.

The remainder of Chapter 1 presents formal definitions for cellular automata, and the combined, transient and manufacturing fault models. The chapter ends with Theorem 1 which shows that the (pure) manufacturing and (pure) transient fault models are not directly comparable.

Chapter 2 presents the result, as mentioned above, that monotone[4] binary cellular automata (with some natural restrictions) are unable to remember a bit in the presence of combined-faults on the two-dimensional grid $\mathbf{Z}^2$. Following this negative result, we show how to use an additional dimension of space to enable any automaton to tolerate combined faults, given that it tolerates transient faults without the additional dimension. The construction applies to our most general definition of cellular automata and to any stable trajectory, not just the "all zeros" and "all ones" trajectories.

With a desire to find cellular automata on two-dimensional surfaces, and in light of the negative result in Chapter 2, in Chapter 3 we study cellular automata embedded in the hyperbolic plane. Section 3.1 gives a number of simple graph-theoretical definitions and proves some simple results that will be useful for the major results later in the chapter. In Sections 3.2 and 3.3, for both the combined and transient fault models, we give the complete classification of majority-rule automata[5] able to remember a bit, and whose cellular spaces are the regular hyperbolic tessellations. Our results are presented for a class of tessellations much more general than just the regular tessellations.

Chapter 4 gathers together some results that are more quantitative in nature. In Section 4.1, we examine the interplay of fault-rates and cell-neighborhood size. For any given bound on cell-neighborhood size, we establish asymptotically closely agreeing (to within a logarithmic factor) lower and upper bounds on the distance of the fault-rate from 1/2 that ensures no automata (binary, but not restricted to monotone) can remember a bit. As a corollary, we give the first

[4]A function f on n arguments is monotone if $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$ whenever $x_1 \leq y_1, \ldots, x_n \leq y_n$.

[5]Majority-rule automata correct errors by majority-voting on their neighbor cells. See Definition 6 at the beginning of Section 3.2.

result that shows automata with sufficiently large cell-neighborhood sizes can tolerate transient (and combined) fault rates arbitrarily close to 1/2. Section 4.2 gives a construction of an infinite family of finite-sized cellular automata using expander graphs. The constructed automata have an ability to remember a bit for an (asymptotically) optimal length of time—in the presence of combined faults, with high probability, they remember a bit for a time that grows exponentially in the number of cells.

Chapter 5 describes two additional fault models: the delayed-repair fault model, and the periodic fault model. Although seemingly different, we show that these variant fault models are actually equivalent (in a precise sense defined in the chapter) to the transient fault model.

The "Addressing Schemes" appendix is concerned with the existence (and construction) of particular finite edge-colorings for the tessellations described in Chapter 3. These graph edge-colorings are needed in the proof of transient fault tolerance for a subset of the hyperbolic tessellations. They provide a way to assign addresses that have especially useful properties to cells. Although the results in this part are interesting in their own right, they are somewhat tangential to the rest of the thesis, and so we have put them in the appendix.

1.1 Automaton Model

Our definition of automata and their associated notation is similar to those given by Toom in [30], although we have chosen to be slightly less general than Toom.

Traditionally the cells in a cellular automaton are located on the d-dimensional grid lattice $\mathbf{Z}^d$. We will often deal with more general arrangements and so we simply refer to the set of cells as the *medium*. The set of locations of cells are denoted by $\mathbf{M}$ which is assumed to be countable. We follow the convention that $\mathbf{Z}^+ = \{0, 1, \ldots\}$. We define $\mathbf{L} = \mathbf{M} \times \mathbf{Z}^+$ to be the space-time version of the medium and we refer to this as the *space-time lattice* or simply *lattice*. To help avoid confusion, we will usually refer to elements of $\mathbf{M}$ as *cells* and to elements of $\mathbf{L}$ as *points*. A point is a cell at a particular time. For $a \in \mathbf{L}$, let $\underline{a}$ denote the projection of a onto $\mathbf{M}$, and let $t(a)$ denote the projection of a onto $\mathbf{Z}^+$. For $A \subseteq \mathbf{L}$, let $\underline{A} = \bigcup_{a \in A} \{\underline{a}\}$, and let $\underline{f}(\cdot)$ denote $f(\underline{\cdot})$ where f is any function which maps to subsets of $\mathbf{L}$. Let $\mathbf{L}_n = \{a \in \mathbf{L} \,|\, t(a) = n\}$ be the subset of lattice points with common time n. We call $\mathbf{L}_0$ the *boundary* and $\mathbf{L} \setminus \mathbf{L}_0$ the *interior*.

Associated with each point $a \in \mathbf{L}$ is a finite set X_a. Elements of X_a are called *states* and we require that $X_a = X_b$ for all a, b where $\underline{a} = \underline{b}$. As a convenience, for $a \in \mathbf{M}$, define $X_a = X_{(a,1)}$. Usually $X_a = \{0, 1\}$. For $A \subseteq \mathbf{L}$ we define $X_A = \prod_{a \in A} X_a$ and call elements of X_A *configurations*. Without mention of a specific subset, a configuration is assumed to be over the entire lattice. We sometimes denote $X_{\mathbf{L}}$ as simply X.

Associated with each cell $a \in \mathbf{M}$ is a finite set of cells $U_{\mathbf{M}}(a)$ called a's *input-cells*. With $U_{\mathbf{M}}(a)$ fixed, for each point $a \in \mathbf{L} \setminus \mathbf{L}_0$ we define a set $U_{\mathbf{L}}(a) =$

$\{(b, t(a)-1) : b \in U_{\mathbf{M}}(\underline{a})\}$ called a's *input-points*. For $a \in \mathbf{L}_0$, $U_{\mathbf{L}}(a) = \emptyset$. When it is clear from the context, we will sometimes drop the qualifying subscript from U. We sometimes refer to $U(a)$ simply as a set of *inputs* or an *input-set*. For $U = U_{\mathbf{M}}$ and $A \subseteq \mathbf{M}$, or $U = U_{\mathbf{L}}$ and $A \subseteq \mathbf{L}$, we define $U^0(A) = A$, $U^k(A) = U(U^{k-1}(A))$ for $k > 0$, and $U^\infty(A) = \bigcup_{k \geq 0} U^k(A)$. Note that for every $a \in \mathbf{L}$, there is a minimum k such that $U^k_{\mathbf{L}}(a) = \emptyset$ which we call the *depth* of a point. We generally assume that $U_{\mathbf{L}}(a) \neq \emptyset$ for all $a \in \mathbf{L} \setminus \mathbf{L}_0$. With this assumption, the depth of a point is equivalent to its time-projected value (i.e. for all $a \in \mathbf{L}$, we assume $U^{t(a)-1}_{\mathbf{L}}(a) \neq \emptyset$, and $U^{t(a)}_{\mathbf{L}}(a) = \emptyset$).

For each $a \in \mathbf{M}$ there is a transition function $\phi_a : X_{U(a)} \to X_a$. For convenience, we define $\phi_a = \phi_{\underline{a}}$ for all $a \in \mathbf{L}$. A configuration $x \in X_{\mathbf{L}}$ is called a *trajectory* if $x_a = \phi_a(x_{U(a)})$ for all $a \in \mathbf{L} \setminus \mathbf{L}_0$. A configuration of the boundary $x \in X_{\mathbf{L}_0}$ determines a trajectory, and we denote it $\mathsf{tr}(x)$.

Note that the functions $U_{\mathbf{M}}$ and $U_{\mathbf{L}}$ can be used to define directed multigraphs on vertex sets $\mathbf{M}$ and $\mathbf{L}$ respectively. For example, we can say there is an edge directed from cell a to cell b if $b \in U_{\mathbf{M}}(a)$. Thus, we think of edges from a pointing to the cells on which a's transition function ϕ_a depends. Note that this convention is the opposite of the convention used in circuits where edges are thought to point in the direction of electrical current or information flow.

1.2 Fault Models

A major objective of this thesis is to explore new models of incorrectness for cellular automata. We begin with some formal definitions for these models.

Every finite $A \subset \mathbf{M}$ corresponds to a cylinder set $\mathsf{cyl}_{\mathbf{M}}(A)$ on the product space $2^{\mathbf{M}}$ where $\mathsf{cyl}_{\mathbf{M}}(A)$ is defined by

$$\mathsf{cyl}_{\mathbf{M}}(A) = \{M \in 2^{\mathbf{M}} : A \subseteq M\}.$$

We denote by $F_M \subseteq \mathbf{M}$ the set of *manufacturing faults*. Let α be a parameter $0 \leq \alpha \leq 1$ called the *manufacturing fault rate*. Let $\Sigma_{\mathbf{M}}$ denote the σ-algebra generated by all cylinder sets. For every value α, the distribution of manufacturing faults is defined by the probability measure μ_α on measurable space $(2^{\mathbf{M}}, \Sigma_{\mathbf{M}})$. The measure μ_α is uniquely defined (see Ch. 3, Sec. 4 of Kolmogorov's book [19]) by requiring that for every finite $A \subset \mathbf{M}$,

$$\mu_\alpha(\mathsf{cyl}_{\mathbf{M}}(A)) = \alpha^{|A|}.$$

This is just a formal way of saying that manufacturing faults occur independently with probability α at each cell of the medium.

Similarly, by substituting $\mathbf{L}$ for $\mathbf{M}$ we get the measurable space $(2^{\mathbf{L}}, \Sigma_{\mathbf{L}})$ along with cylinder sets $\mathsf{cyl}_{\mathbf{L}}(A)$ where A is a finite subset of $\mathbf{L}$. We call $F_T \subseteq \mathbf{L}$ the set of *transient faults* and the *transient fault rate* is described by the parameter $0 \leq \beta \leq 1$. The distribution of transient faults is described by the probability

measure ν_β on $(2^{\mathbf{L}}, \Sigma_{\mathbf{L}})$. The measure ν_β is uniquely defined by requiring that for every finite $A \subset \mathbf{L}$,

$$\nu_\beta(\mathsf{cyl}_{\mathbf{L}}(A)) = \beta^{|A|}.$$

Again, this formal definition simply says that transient faults occur independently at every point in the lattice with probability β.

Let $(\mu_\alpha \times \nu_\beta)$ be the product measure of μ_α and ν_β. Let C_y be the set of configurations on $\mathbf{L}$ with initial conditions identical to trajectory y. When both fault rates α and β are non-zero, we say that the automaton is operating under the *combined fault model*. In order to unify our sets of faults F_M and F_T, we define a *combined fault set* $F_C = \{(a,t) \in \mathbf{L} : a \in F_M,\, t \in \mathbf{Z}^+\} \cup F_T$. For brevity, we will sometimes slightly abuse notation by referring to F_M as though it were $\{(a,t) \in \mathbf{L} : a \in F_M,\, t \in \mathbf{Z}^+\}$; however, the intended meaning will always be clear from context. For any configuration x, we define the *set of failures* (with respect to x) as $D_x = \{a \in \mathbf{L} : x_a \neq \phi_a(x_{U(a)})\}$. There are many possible notions of "error," but we choose to define an error as an event that occurs with respect to a desired trajectory. Let y be a trajectory and x a configuration. The *set of errors* is $E_{x,y} = \{a \in \mathbf{L} : x_a \neq y_a\}$. Notice that the set of errors is defined with respect to a fixed trajectory, but the set of failures is independent of a trajectory. A failure at a point implies a fault, but a fault does not necessarily imply a failure.

Definition 1. A trajectory y is *stable for combined faults* if

$$\limsup_{\substack{\alpha\to 0\\ \beta\to 0}} \sup_{a\in\mathbf{L}} (\mu_\alpha \times \nu_\beta)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}) = 0.$$

For the combined fault model we let $\varepsilon = \alpha + \beta - \alpha\beta$ (it follows that $0 \leq \varepsilon \leq 1$, and that $\varepsilon \to 0$ if and only if $\alpha \to 0$ and $\beta \to 0$) and call ε the *combined fault rate*. The advantage of ε is that it bounds the probability of a fault occurring at any point in the lattice with a single parameter. We will often refer to ε when speaking of the combined fault model. (The limit as α and β tend to zero always exists by the monotonicity and non-negativity of the supremum that follows.)

Definition 2. A trajectory y is *stable for manufacturing faults* if

$$\limsup_{\alpha\to 0} \sup_{a\in\mathbf{L}} (\mu_\alpha \times \nu_0)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}) = 0.$$

Definition 3. A trajectory y is *stable for transient faults* if

$$\limsup_{\beta\to 0} \sup_{a\in\mathbf{L}} (\mu_0 \times \nu_\beta)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}) = 0.$$

When the fault model and fault-rate are understood, we will often write Pr in place of $(\mu_\alpha \times \nu_\beta)$. For example, to represent the probability that point a is in error, given an initial configuration y, and under the combined fault model with faults rates α and β, we will generally write simply $\Pr[a \in E_{x,y}]$ rather than $(\mu_\alpha \times \nu_\beta)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\})$.

The traditional incorrectness models, as described in [30, 11], correspond to our transient fault model, and we have many results which use this model for incorrectness. Our manufacturing faults are a source of incorrectness not previously studied. These types of faults describe a new fault model on their own (the manufacturing fault model), but we mainly consider them in combination with transient faults—the combined fault model.

The "$\exists x \in C_y$" in the above definitions represents our choice of using an adversarial model—the adversary is all-knowing and can choose not to violate a cell's deterministic transition function when given the opportunity to do so. If the adversary is removed, then we have a *probabilistic automaton* where the sets of faults and failures correspond exactly. For example, we can make probabilistic versions of Definitions 1 through 3 by changing $D_x \subseteq F_C$ to $D_X = F_C$. This gives the automaton a source of random coin flips and it is quite possible that the automaton relies on this randomness to function correctly. With an adversary, failures are always harmful since the adversary is all-knowing and only uses faults as opportunities to cause harm. For proving positive results, the adversarial model is therefore stronger than the probabilistic model in the sense that stability with an adversary implies stability without an adversary. For negative results, the opposite is true. Historically, it has proven to be very difficult to prove negative results without an adversary (see however [23] and our Theorem 30). The majority of our results are for fault models with an adversary. (In Toom's definition for transient-fault-model stability [30], the power of the adversary is represented by taking a supremum over a set of measures which represent all possible strategies for the adversary. In our Definition 1, the adversary's choice is represented by the "there exists" clause within the set of events statement. It is not hard to show that the two definitions are equivalent.)

We will often be concerned with the special case of monotone binary automata. The natural trajectories to consider for such automata are the "all zeros" and "all ones" trajectories which we denote $\mathbf{0}$ and $\mathbf{1}$ respectively. We say that any automaton which has both $\mathbf{0}$ and $\mathbf{1}$ as stable trajectories is able to *remember a bit* under the particular fault model for which both trajectories are stable. We say that an automaton is *tolerant of combined faults* if it is able to remember a bit under the combined fault model. Similarly, we say that an automaton is *tolerant of manufacturing faults* or is *tolerant of transient faults* if it can remember a bit with manufacturing faults or transient faults respectively. If an automaton cannot remember a bit under a particular fault model, then we say that it is *intolerant* of faults under that fault model.

We have defined the stability of a trajectory by considering the limiting be-

havior of the worst possible point as the fault-rate approaches zero. It will sometimes be useful to say that an automaton can or cannot remember a bit for a given fault-rate under some particular fault model.

Definition 4. Assuming a particular fault-model and a particular trajectory, we say that an automaton is *tolerant of fault-rate* ε if the probability that any point is in error is bounded below $1/2$ by some fixed $\delta > 0$.

As an example, consider the combined fault model. The trajectory $\mathbf{0}$ is stable for fault-rate ε if there exists $\delta > 0$ such that

$$\sup_{a \in \mathbf{L}} (\mu_\alpha \times \nu_\beta)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}) \leq 1/2 - \delta.$$

Given a formal definition of stability (as in Definitions 1, 2 and 3), it should be clear how to make Definition 4 mathematically precise as we did in the above for combined fault-tolerance.

The following result shows that manufacturing faults are not either strictly worse (with respect to the stability of an automaton), or better than transient faults. This disproves the notion that the transient, manufacturing, and combined fault models are in a strict hierarchy with manufacturing faults either in the middle, or equivalent to, combined faults. It is also part of the reason that we focus on combined faults, rather than pure manufacturing faults.

Theorem 1. *Even restricted to monotone binary automata, the pure manufacturing and transient fault models are not comparable in the following sense: there exist monotone binary automata which are tolerant under manufacturing faults but not transient faults and there exist monotone binary automata which are tolerant under transient faults but not manufacturing faults.*

Proof. We give examples of each situation. Consider any finite or infinite collection of cells whose transition function is just the identity function of the cell itself. For manufacturing faults, as long as the fault-rate is less than $1/2$, then each cell has a probability of being in error of less than $1/2$ at every timestep. But for the transient fault model, with probability one, any chosen cell will be permanently in error after a finite number of transition steps. Even under the weaker assumption of having no adversary, the probability of being in one state or the other approaches $1/2$.

In [28], Toom describes a very simple automaton able to remember a bit under the transient fault model. The monotone binary automaton has $\mathbf{Z}^2$ as the medium. The transition rule at each cell is a majority vote of itself and the cells immediately to the north and east. We refer to this rule as *Toom's Rule*. Now consider a modified version of Toom's Rule. At each timestep, every cell takes a majority vote on cells two steps north and one step east, one step north and one

Figure 1.1: Toom's Rule and a variation of Toom's Rule.

step east, and two steps east and one step north. This is essentially the same as Toom's Rule except that the state of the entire system is shifted one place to the south and west at each timestep. Figure 1.1 illustrates the three cells that a single cell depends upon for both Toom's Rule and the variation. It should be clear that this system is still tolerant of transient errors, however one can verify this using a simple criterion of Toom as described in [29]. Now consider the behavior of this modified Toom's Rule in the case of pure manufacturing faults. With probability one, there is a square island of four cells with manufacturing faults n steps to the north and n steps to the east. The adversary sets these four cells into a state of perpetual error. After n timesteps, this island of error will have spread to the origin, which will remain permanently in error. □

Chapter 2

Combined Faults on $\mathbf{Z}^2$

2.1 Memory on $\mathbf{Z}^2$

All of the automata we consider in this section are monotone and binary. We say that an automaton can remember a bit under the given fault model if both the trajectories $y = \mathbf{0}$ and $y = \mathbf{1}$ are stable. The monotone and binary restrictions combined with the problem of remembering an unencoded[1] bit are useful because they give the adversary an optimal strategy when faults occur: the adversary always sets the state of an at-fault cell to the opposite of the initial common state.

We begin with definitions of implicants and co-implicants that are slightly non-standard in form, but not in spirit. Let $a_1, \ldots, a_n$ be the variables of a Boolean function f and suppose that $f(a_1, \ldots, a_n) = 1$ whenever $a_i = 1\ \forall i \in A \subseteq \{1, \ldots, n\}$. Then A is called an *implicant* of f. Similarly, if $f(a_1, \ldots, a_n) = 0$ whenever $a_i = 0\ \forall i \in A \subseteq \{1, \ldots, n\}$, then A is called a *co-implicant* of f.

Lemma 2. *Let $f(a_1, \ldots, a_n)$ be a monotone Boolean function and let $A, B \subseteq \{1, \ldots, n\}$ such that $A \cup B = \{1, \ldots, n\}$. Then at least one of A or B is an implicant or a co-implicant for f.*

Proof. Set $a_i = 1$ for all $i \in A$ and set $a_j = 0$ for all $j \in \{1, \ldots, n\} - A$. If $f(a_1, \ldots, a_n) = 1$, then A is an implicant (since f is monotone). Otherwise $f(a_1, \ldots, a_n) = 0$ which implies $\{1, \ldots, n\} - A$ is a co-implicant (again, since f is monotone). But since $\{1, \ldots, n\} - A \subseteq B$, by the definition of co-implicants, B is also co-implicant. By duality, we also conclude that either A is a co-implicant or B is an implicant. □

[1] The only "encoding" going on here is that of replication. It is possible that other trajectories other than "all zeros" and "all ones" might be stable, even when restricted to monotone binary cellular automata, but then determining the state of the automaton at any given time could not by done by simply looking at the state of an arbitrary single cell at an arbitrary timestep.

Let $C \subseteq \mathbf{R}^2$ be such that $\alpha x + \beta y \in C$ for every $x, y \in C$ and positive scalars α and β. Then C a called a *cone*.

Lemma 3. *For every monotone transition rule $\phi = (x_1, \ldots, x_n, f)$ on lattice medium $\mathbf{Z}^2$, there exist two cones $C_1, C_2 \subseteq \mathbf{R}^2$ which satisfy the following:*

1. *Each cone contains no complete line in its interior.*
2. *The interiors of C_1 and C_2 are disjoint.*
3. *$\{i \in \{1, \ldots, n\} : x_i \in C_1\}$ and $\{j \in \{1, \ldots, n\} : x_j \in C_2\}$ are either both implicants, or both co-implicants for f.*

Proof. The key to our proof is the construction of a cyclic sequence of pairs of sets of neighbor vector indices. Consider the set of all normalized, non-zero displacement vectors in $x_1, \ldots, x_n$ unioned with their negatives, and let $z_1, \ldots, z_m$ be an ordered version of this set, sorted in increasing, counterclockwise order, from some arbitrarily chosen vector in the set.

By our construction, m is even and at most $2n$, but it could be as small as 2 if all of the input vectors point in the same direction. Let $C(a, b)$ be the set of indices of neighbors in $x_1, \ldots, x_n$ belonging to the closed and convex cone defined by vectors a, and b where the boundary of the cone is from vector a to vector b in counterclockwise order. For example, $C((-1, 0)^t, (1, 0)^t)$ is the set of all points $\{(x, y)^t \in \mathbf{R}^2 : y \leq 0\}$. We define an initial finite sequence $\mathcal{S}_0 = \{s_i\}_{0 \leq i < m/2}$ as follows:

$$
\begin{aligned}
s_{4i+0} &= (C(z_i, z_{i+m/2}), C(z_{i+m/2}, z_i)) \\
s_{4i+1} &= (C(z_{i+1}, z_{i+m/2}), C(z_{i+m/2}, z_i)) \\
s_{4i+2} &= (C(z_{i+1}, z_{i+m/2}), C(z_{i+m/2+1}, z_i)) \\
s_{4i+3} &= (C(z_{i+1}, z_{i+m/2+1}), C(z_{i+m/2+1}, z_i)).
\end{aligned}
$$

We now extend $\mathcal{S}_0$ into an infinite sequence $\mathcal{S}$ by starting with $\mathcal{S}_0$ and then repeating $\mathcal{S}_0$, but with its pairs reversed from left to right. The sequence defined now has length $4m$. We then repeat this whole sequence indefinitely. (The simplicity of the construction is well illustrated by the example in Figure 2.1. In this example, the vectors have been scaled to unit length and drawn on the unit circle for clarity and because the construction of $\mathcal{S}$ depends only on angles between $x_1, \ldots, x_4$.)

The sequence $\mathcal{S}$ has a number of useful properties which follow easily from its construction. For $s_i, s_{i+1} \in \mathcal{S}$ where $s_i = (A_i, B_i)$ and $s_{i+1} = (A_{i+1}, B_{i+1})$, the following are facts:

(1) $A_i \cup B_i = \{1, \ldots, n\}$.

(2) The interiors of the cones which span A_i and B_i are non-overlapping.

(3) The cones which include A_i and B_i each do not contain a complete line in their respective interiors.

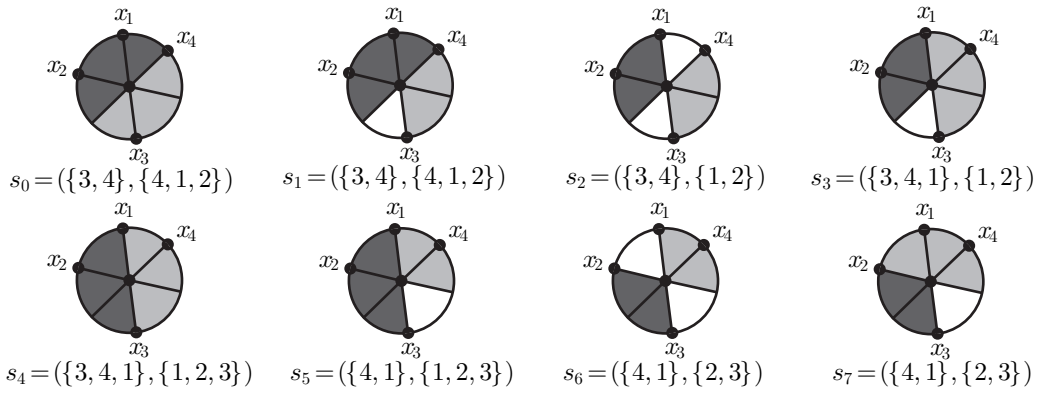

Figure 2.1: An example of how the sequence $\mathcal{S}$ is created for a transition rule with displacement vectors $x_1, \ldots, x_4$. In this example, the sequence repeats every 24 entries.

(4) $\mathcal{S}$ is periodic with period $4m$.

(5) Either $A_i = A_{i+1}$ or $B_i = B_{i+1}$.

(6) A subset relation holds between A_i and A_{i+1} (that is, either $A_i \subseteq A_{i+1}$, or $A_{i+1} \subseteq A_i$), and a subset relations holds between B_i and B_{i+1}.

For the sake of a contradiction, assume that $\mathcal{S}$ contains no entry (A, B) for which A and B are either both implicants or both co-implicants. By fact (1) and Lemma 2, A_i or B_i is an implicant or a co-implicant. Now consider any consecutive pair s_i and s_{i+1}. Then it is not possible that A_i and B_{i+1} are either both implicants or co-implicants and similarly for A_{i+1} and B_{i+1}. This follows easily from facts (5) and (6), and from our current working assumption. The implication is that between any two consecutive pairs of entries (A_i, B_i) and (A_{i+1}, B_{i+1}), it is not possible that one of A_i or A_{i+1}, and B_i or B_{i+1} are implicants or co-implicants. Therefore, if A_i is an implicant, then all other A_j $(j \geq i)$ are not co-implicants, and all other B_j are not implicants. But, by fact (4) this is impossible since $\mathcal{S}$ is periodic. Therefore, there exists a pair $s_i = (A_i, B_i)$ where either both A_i and B_i are both implicants or both co-implicants. The rest of the Lemma follows from facts (2) and (3). □

In [30], Toom gave a complete criterion for determining if a monotone binary cellular automaton in $\mathbf{Z}^d$ is able to remember a bit with transient faults present. The existence of an optimal strategy for the adversary is key to his proof. In contrast to Toom's criterion, which shows that many monotone binary transition rules can remember a bit in $\mathbf{Z}^2$, we have:

Theorem 4. *No cellular automaton on the simple two-dimensional lattice $\mathbf{Z}^2$ with a monotone binary transition rule can remember both the "all zeros" and "all ones" initial state when combined faults are present.*

Proof. Let $\phi = (x_1, \ldots, x_n, f)$ be the transition rule and let C_1 and C_2 be the two cones with the properties specified in Lemma 3 and let $A = C_1 \cap \{x_1, \ldots, x_n\}$ and $B = C_2 \cap \{x_1, \ldots, x_n\}$. By conclusions 1 and 2 of Lemma 3, there exists a line l passing through the origin that does not intersect the interiors of either C_1 or C_2. By conclusion 3 of Lemma 3, C_1 and C_2 are either both implicants or both co-implicants. Let $e = 1$ if both are implicants, and $e = 0$ otherwise. Assume the initial state of each cell is the complement $\overline{e}$ of e.

Let $r = \max_{i \in \{1,\ldots,n\}} \|x_i\|$ be the radius of the transition rule. Let $\ldots, q_{-1}, q_0 = (0,0), q_1, \ldots$ be points equally spaced at distance $7r$ along l. For $i \in \mathbf{Z}$, let Q_i be the set of all points of $\mathbf{Z}^2$ within distance $3r$ of q_i. Then (1) the sets $R_i = Q_{-i} \cup Q_i$ are disjoint for distinct values of $i \geq 1$, and (2) for any $i \geq 1$, if all points in the convex hull of R_i are set to e, they will form a self-sustaining island of errors.

Let E_i be the event that all points of R_i have manufacturing faults. Since the R_i are disjoint, the E_i are independent, and they all have the same strictly positive probability (depending on α and r, but independent of i). Thus with probability 1, at least one of the events E_i occurs.

Let i be such that E_i occurs. Let S_i be the set of points in the convex hull of R_i but not in R_i itself. For $t \geq 1$, let $F_{i,t}$ be the event that all points of S_i have transient faults simultaneously at time t. The $F_{i,t}$ are independent, and they all have the same strictly positive probability (depending on β, r and i, but independent of t). Thus with probability 1, $F_{i,t}$ occurs for some $t \geq 1$, and the origin will be in error at all later times. □

2.2 Additional Dimensions Are Useful

Theorem 5. *Any cellular automaton which is transient-fault tolerant can be made combined-fault tolerant by adding a dimension.*

The truth of this theorem is easy to see from a simple construction. Take any cellular automaton and "stack" a countably infinite number of copies of it, one "on top" of another. Now, modify the transition rule so that every cell gets its inputs from its neighbors in the plane copy above it. Figure 2.2 illustrates this construction for Toom's rule starting with a two-dimensional cellular automaton.

As mentioned in the Introduction, we can use this three-dimensional version of Toom's rule to perform reliable computation with combined faults in five dimensions. Of course, the construction is not limited to automata in any particular cellular space having any particular limitations on its transition rule (e.g. monotone) or set of states (e.g. binary). In [13][2], Gács was able to construct a one-dimensional cellular automaton able to perform arbitrary computation with

[2]See Gács website http://www.cs.bu.edu/fac/gacs/recent-publ.html for the most up-to-date version of this paper.

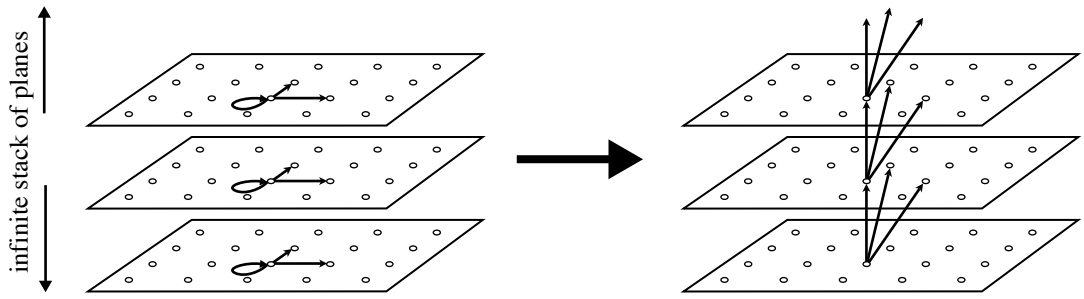

Figure 2.2: The construction of a combined-fault tolerant cellular automaton in three dimensions from a transient-fault tolerant cellular automaton in two dimensions.

transient faults present. This amazing feat, which disproved the discrete time version of the "positive rates conjecture" (a folk-conjecture of the condensed-matter physics community) requires a very complex "transition rule."[3] Of course, with combined faults, any one-dimensional automaton will forget its initial state because every cell is trapped in an isolated island between two islands of manufacturing faults. Using Gács' one-dimensional cellular automaton and Theorem 5, it is therefore possible to construct a combined-fault tolerant automaton on the two-dimensional grid.

[3]This automaton required over 120 pages of journal space to describe and prove.

Chapter 3

Memory on Hyperbolic Tessellations

The negative result for automata in two dimensions, as presented in Section 2.1, can be viewed as a serious problem because there are compelling reasons to remain in two-dimensional space (see [12] for example). The strongest argument might be that for an arbitrarily large computer, power consumption and heat dissipation is a major concern. In two dimensions, the third dimension of space can be used for power delivery and heat removal. In our physical world, a three-dimensional automaton of arbitrary size can therefore not be built. Furthermore, [26] Pippenger has shown that no rule on a finite dimensional hypercubic lattice, whose transition rule is monotone and self-dual[1] (majority is an example of such a rule), and is invariant under inversion, can remember a bit in the presence of transient faults. In light of Pippenger's result, and by our desire to study automata on two-dimensional surfaces, we study automata on the hyperbolic plane.

This chapter presents a number of results for automata where the medium requires the hyperbolic plane for its embedding. Section 3.1 presents a number of simple graph theoretic definitions and simple results that will be used frequently in Sections 3.2 and 3.3, and elsewhere in the thesis. A good way to view the types of graphs we are interested in is as infinite-sided polyhedra. Many of our results are for graphs that are trees, and these can also be viewed as polyhedra by allowing faces to be infinite[2]. See Figure 3.1 (page 21) and Figure 3.2 (page 21) for examples of some regular tessellations. Definition 5 defines the set of graphs that we are most interested in, while Theorems 10 and 12 show some useful properties of these graphs that will be used as a basic tool in many other results. The planar graphs to which the results of this chapter apply were motivated by the regular

[1] A Boolean function is self-dual if, for all input-value combinations, the value of the complement of the function equals the value of the function with its inputs complemented.

[2] In the Appendix, we define polyhedra as a subset of graphs described in this section, and the Appendix is the only section where we use the term "polyhedra" in a technical way.

tessellations of the hyperbolic plane, but in fact our results do not depend on the symmetry inherent to regular tessellations. It is possible to skim over Section 3.1 at first, and then refer to it as needed later.

3.1 Definitions and Some Graph Theory

In graph theory, there is sometimes disagreement in terminology of even the most basic notions and therefore, to avoid confusion, we begin with some simple definitions. Many of our definitions follow those given in [7]. Although we normally assume that graphs are undirected, we will occasionally find it convenient to use directed graphs and therefore we define them as well. We denote both undirected and directed graphs by the ordered pair (V, E) where V is the set of vertices and E is the set of edges. In an undirected graph, edges are sets of two vertices. We will often denote edge $\{u, v\} \in E$ as uv, or equivalently vu. In a directed graph, edges are ordered pairs of vertices. We will often denote a directed edge $(u, v) \in E$ as uv. In both cases, we assume graphs are *simple*, meaning that self-edges (or loops) and multiedges are not allowed. For a graph G, we let $V(G)$ and $E(G)$ denote the set of vertices and edges respectively. In an undirected graph, an edge $e = \{u, v\}$ is said to have u and v as its *endvertices* and e is said to be *incident* on u and v. If $\{u, v\} \in E$, then u is said to be *adjacent* to v and u and v are said to be *neighbors*. The *neighborhood* of $v \in V(G)$ is the set of neighbors of v and is denoted $N(v)$. For a directed graph G, we say that u is adjacent to v if and only if $(u, v) \in E$. For vertex u, we define the *out-neighborhood* $N^+(u)$ to be the set of vertices adjacent to u. Similarly, we define the *in-neighborhood* $N^-(u)$ to be all the vertices that u is adjacent to. For a directed graph, the *neighborhood* of a vertex u is $N(u) = N^+(u) \cup N^-(u)$. For a directed or an undirected graph, the *degree* of a vertex u is defined $d(u) = |N(u)|$. For a directed graph, we also define the *out-degree* and *in-degree* of vertices defined by $d^+(u) = |N^+(u)|$ and $d^-(u) = |N^-(u)|$ respectively. In all our graphs, we assume that the set of vertices and edges are countable. A *locally finite* graph is a graph where the degree of each vertex is finite. In general, our graphs are assumed to be locally finite.

A *ray* P is an undirected graph whose vertex set can be put into a sequence $\{v_i\}$ indexed by $\mathbf{Z}^+ = \{0, 1, \ldots\}$ such that for all i, $(v_i, v_{i+1}) \in E(P)$ and $v_i \neq v_j$ when $i \neq j$. A *double-ray* is similar to a ray except that its vertices can be put into a sequence $\{v_i\}$ indexed by $\mathbf{Z}$. A *finite path* of length n has $n + 1$ vertices with the property that its vertices can be put into a sequence $\{v_i\}$ indexed by $\{0, \ldots, n\}$. The sequence of vertices has the property that $(v_i, v_{i+1}) \in E(P)$ for $0 \leq i < n$ and $v_i \neq v_j$ when $i \neq j$. Rays, double-rays and finite paths are all considered to be types of *paths*. The *length* of a path, denoted $|P|$, is the number of edges in P.

A *walk* is a sequence of vertices $W = \ldots, v_k, v_{k+1}, v_{k+2}, \ldots$ on either a directed or an undirected graph G, with the following property: for every pair of successive vertices v_i, v_{i+1} on the walk, v_{i+1} is adjacent to v_i in graph G. Similar to paths,

walks can be finite, singly infinite, or doubly infinite. However, unlike paths, walks can self-intersect.

The *distance* between two vertices $a, b \in V(G)$ is defined by the length of a shortest path connecting those vertices and is denoted $\mathsf{dist}(a, b)$. If there is no path between a and b, then we define $\mathsf{dist}(a, b) = \infty$. The distance between a vertex $a \in V(G)$ and a subset $B \subseteq V(G)$ is defined as

$$\mathsf{dist}(a, B) = \mathsf{dist}(B, a) = \min_{b \in B} \mathsf{dist}(a, b).$$

Similarly, if A is also a subset of vertices, then

$$\mathsf{dist}(A, B) = \mathsf{dist}(B, A) = \min_{a \in A,\, b \in B} \mathsf{dist}(a, b).$$

There will often be a distinguished vertex $a^* \in V(G)$ called *the origin* of the graph. When an origin a^* is understood, we might use the shorthand notation

$$\mathsf{dist}(a) = \mathsf{dist}(a^*, a).$$

Let Γ be a plane graph representation of the abstract graph G. A *vertex accumulation point (VAP)* is a point x in the plane for which every neighborhood of x contains infinitely many vertices of Γ. In the following, we assume a fixed embedding in the plane for G and do not generally distinguish between G and its planar embedding. We say that G has no vertex accumulation points or is *VAP-free* if it has an embedding Γ that contains no vertex accumulation points. For our purposes, an important property of a planar embedding is that it fixes an ordering of edges around every vertex and partitions the space occupied by the the graph into a set of faces. We will not often refer to the VAP-free property directly, but it will be assumed in all of our graphs to ensure that finite regions of an embedding in the plane contain a finite number of graph components.

When a vertex has been identified as the origin in a connected graph, we can refer to parent and child relationships between vertices. The origin is viewed as root ancestor of all vertices and all other vertices are descendants of the origin. The set of all vertices at a fixed distance from the origin is called a *shell* or equivalently a generation of *cousins*. We denote by $\mathfrak{S}_n$ the shell with vertices distance n from the origin. Thus, shells partition the set of vertices. If vertices a and b share an edge, then $|\mathsf{dist}(a) - \mathsf{dist}(b)|$ is either zero or one. If it is one, then the vertex closer to the origin is called the *parent* while the more distant vertex is called the *child*. An edge between a parent on shell n and child is called a *descendant-edge* on shell n (or of order n). If the vertices are the same distance, say n, from the origin, then a and b share a *cousin-edge* on shell n or of order n. Therefore, the set of edges is partitioned into the descendant-edges and cousin-edges.

We will frequently encounter sets of objects in the plane which have a natural cyclic order. For example, a planar embedding of a graph fixes a cyclic order

for the set of neighbor vertices about each vertex in the graph. For an object (typically a vertex) appearing along a non-self-intersecting closed loop (or curve) in the plane, we define the *successor* or *object to the right*, to be the next object encountered along the curve moving in a clockwise direction. Similarly, the *predecessor* or *object to the left*, is the next object encountered along the curve moving in a counterclockwise direction. A pair of objects for which the successor or predecessor relation holds are called *successive* objects. When it is understood that a set of objects has a cyclic order, we might use a phrase such as "the second successor" or "the object two to the right" which should be understood to mean the successor of the current object's successor. When a proper (finite) subset of successive objects is given, then there is a unique object whose successor is not in the subset, and we call this the *rightmost* object. Similarly, the *leftmost* object is the object having no predecessor in the current subset.

A *face* of a planar graph is a connected, open region of the plane whose boundary is a simple-cycle[3] of edges and vertices. The *degree* of a face is the number of edges (equivalently vertices) that bound the face. We will sometimes use the phrase "the edges (or vertices) in a face," instead of "the edges (or vertices) that bound a face" even though technically the edges (vertices) are only in the closure of the face. Any point in the open region of the face (and not on the boundary) is in the *interior*, and any point not on the boundary and not in the interior in the *exterior*.

Let G be a plane-embedded, VAP-free graph. We call a vertex a of G *trapped* if a is in the interior of a face P such that $\max_{b \in P} \mathsf{dist}(b) \leq \mathsf{dist}(a)$. We say that a graph G possesses the *strong descendant property* if, with respect to any origin, every vertex has a child. Obviously, any graph with the strong descendant property has an infinite number of vertices and edges.

Lemma 6. *Let G be a plane-embedded, VAP-free graph with the strong descendant property. Then G has no trapped vertices.*

Proof. For the sake of a contradiction, suppose there exists a vertex v trapped within face P. Now v's child must also be in the interior of P since v's child is on shell $\mathsf{dist}(v) + 1$ while every vertex in P is at most on shell $\mathsf{dist}(v)$. Therefore, all of v's descendants will be trapped within P and since v has an infinite number of vertices, P contains an infinite number of vertices in its interior. This implies the existence of a VAP which is a contradiction. □

Definition 5 ($\mathfrak{G}$, $\langle p, q \rangle$, $[p, q]$)**.** Let $\mathfrak{G}$ be the set of plane-embedded graphs which are connected, undirected, locally finite, simple (no self- or multi-edges), and vertex accumulation point free (VAP-free). We frequently refer to certain subsets of graphs in $\mathfrak{G}$ and so we define the following notation. Let $\langle p, q \rangle$ denote the subset of graphs $G \in \mathfrak{G}$ such that:

[3]A simple-cycle is formed from a finite-path with an additional edge connecting the end-vertices of the path.

- the degree of every face in G is at least p.
- the degree of every vertex in G is at least q.
- G possesses the strong descendant property.

To denote a tree-graph conveniently, we write $\langle \infty, q \rangle$, which indicates that all faces in the graph are infinite. Additionally, we define the following companion classes of graphs. Let $[p, q]$ denote the subset of graphs in $G \in \mathfrak{G}$ such that:

- the degree of every face in G is at most p.
- the degree of every vertex in G is at most q.

We write $[\infty, q]$ to denote the class of graphs where faces are permitted to be infinite. Anytime we consider graph classes $[p, q]$, we do not require the strong descendant property and therefore we have dropped it from the list of requirements. Dropping the strong descendant property means that finite graphs can also be included in $[p, q]$ classes.

Related to graph classes $\langle p, q \rangle$ and $[p, q]$, we define constants $p_{\min}$, $p_{\max}$, $q_{\min}$ and $q_{\max}$ to refer to global degree minimum and maximums for faces and vertices respectively. This gives us a convenient way to refer to additional constraints on face and vertex degrees. For example, for the set of graphs $A \subseteq \langle 4, 6 \rangle$, we could additionally restrict A by saying that $p_{\max} = 10$ to indicate that all face degrees are bounded by ten.[4] When there is no global maximum, we define $p_{\min}$ or $q_{\min}$ to be ∞.

Let $\mathcal{P}_{\text{res}}$ be a subset of graphs in $\mathfrak{G}$ where the degree of every face satisfies a restriction "res" where "res" is one of "evn" (even number), "odd" (odd number), "par" (all face degrees have the same parity), "bdd" (all face degrees are bounded by some finite $p_{\max}$), or "fnt" (all face degrees are finite). Similarly, we define $\mathcal{Q}_{\text{res}} \subset \mathfrak{G}$ to denote sets of graphs where the vertex degrees are restricted by res. Note that $\langle p, q \rangle \cap \mathcal{Q}_{\text{fnt}} = \langle p, q \rangle$ since $\langle p, q \rangle$ are assumed to be locally finite. For short, we write $\mathcal{P}_{\text{res1,res2}}$ to denote $\mathcal{P}_{\text{res1}} \cap \mathcal{P}_{\text{res2}}$. For example, $\langle 4, 5 \rangle \cap \mathcal{P}_{\text{par,bdd}} \cap \mathcal{Q}_{\text{odd}}$ denotes the set of all hyperbolic tessellations with faces of degree at least four and of the same parity but bounded, and vertex degrees all finite but of odd degree at least five. Note that when $\langle p, q \rangle$ has the restriction that $p_{\max} = p$ and $q_{\max} = q$, then $\langle p, q \rangle$ contains just a single *regular tessellation* which we denote $\{p, q\}$. Figures 3.1 and 3.2 illustrate some examples of regular tessellations. When $(p-2)(q-2) < 4$, $= 4$, or > 4, the tessellation $\{p, q\}$ is a tessellation of the sphere, Euclidean plane, or hyperbolic plane respectively. See [1], [3, Sec. 4.4], and [6] for a good introduction to the fascinating world of hyperbolic tessellations (their geometry and the groups that generate them).

[4] In general, when we speak of the the vertex (or face) degrees or in a plane-embedded graph being bounded, we mean that a common bound holds for all vertices (or faces).

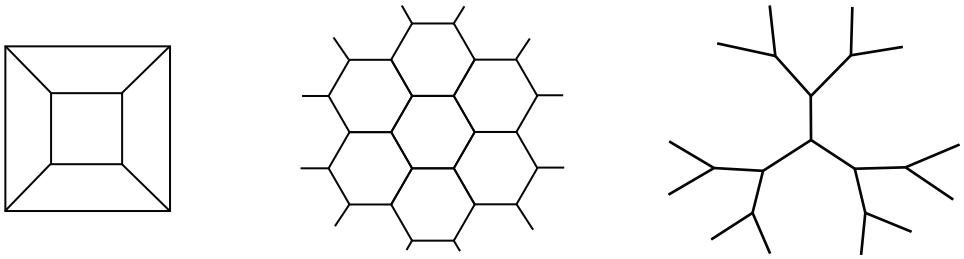

Figure 3.1: From left to right: the regular spherical tessellation $\{4, 3\}$, a portion of the regular Euclidean tessellation $\{6, 3\}$, a portion of the regular tree tessellation $\{\infty, 3\}$.

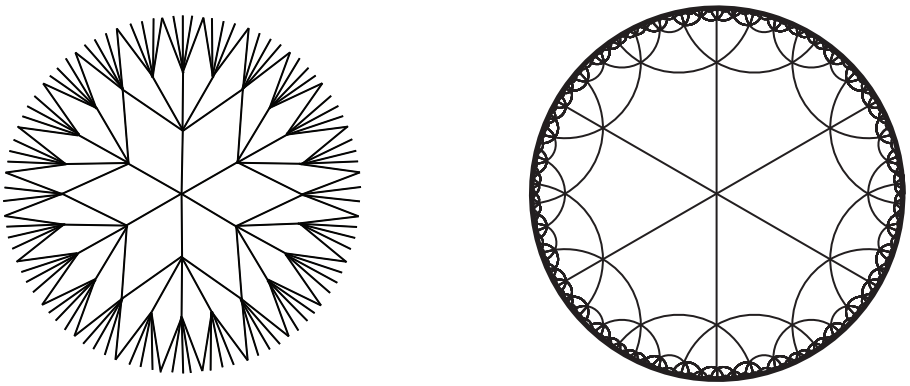

Figure 3.2: On the left: a portion of the regular hyperbolic tessellation $\{4, 6\}$. On the right: the regular hyperbolic tessellation $\{4, 6\}$ projected onto the Poincaré Disk. The projection on the right preserves angles.

We conjecture that for graph G which is simple, undirected, planar, and has an embedding in the plane such that $p_{\min} = 3, q_{\min} = 6$ or $p_{\min} = 4, q_{\min} = 4$, then G has the strong descendant property and there exists a VAP-free embedding of G. If this conjecture were true, then we could remove the strong descendant and VAP-free property from the definition of $\mathfrak{G}$.

The above pairs of values for $p_{\min}$ and $q_{\min}$ above are lower bounds since any of the five convex regular Platonic solids $\{3, 3\}, \{3, 4\}, \{4, 3\}, \{3, 5\}, \{5, 3\}$ provides a counter-example. But trivially, all graphs with the strong descendant property are infinite, so any finite graph has at least one vertex with no descendants.

Consider the nth shell in a locally finite plane-embedded graph. We define an *order n shell boundary* to be a set of disjoint, non-self-intersecting closed curves called *shell boundary curves* with the following properties:

- Every order n descendant edge intersects the shell boundary exactly once and this is a transversal intersection. Furthermore, only one descendant edge intersects the shell boundary at this point.
- An order n shell boundary curve must be intersected by at least one order n descendant edge.

- An order n shell boundary curve is only intersected by order n descendant edges.

When speaking of a set of shell boundaries for different shells, we assume that all shell boundaries are disjoint[5]. Whenever a shell boundary consists of a single curve, we call it a *simple shell boundary.*

Lemma 7. *Suppose that $G \in \mathfrak{G}$ has the strong descendant property. Then all shell boundaries are simple and each shell boundary curve confines all lower order shell boundary curves in its interior. Furthermore, each non-origin vertex v is such that its set of parents occur successively and its set of children occur successively.*

Proof. We prove the lemma by induction on shells. Clearly the only valid shell boundary for the origin is a single closed loop about the origin. The origin has only children, so these are trivially successive. Suppose that the lemma holds for all vertices at or below shell n, and all shell boundaries up to order n. If we can show that shell boundary $n+1$ is simple and that the lemma holds for all vertices on shell $n+1$, then the inductive argument is complete. For the sake of a contradiction, suppose that shell boundary $n+1$ is not simple. Since shell boundary n is simple, all of the vertices at level $n+1$ are connected to the single finite region defined by shell boundary n. This means that any $n+1$ shell boundary curve containing one of vertices $n+1$ must contain them all. Since we assume that more than one $n+1$ shell boundary curve exists, there is at least one curve κ which does not contain shell curve n. By the definition of a shell boundary curve, at least one descendant edge of an order $n+1$ vertex must pass into the interior of κ. By the strong descendant property, there must be an infinite graph within this finite region. This violates the assumption that G is VAP-free, so therefore we must conclude that the $n+1$ shell boundary is simple and contains all objects of lower order.

Consider an order $n+1$ vertex x. For the sake of a contradiction, suppose that two of its parents a and b are not successive. Observe that there is a path leading back to the origin from x through a and another from x through b. So, x is the apex of a face P with the origin which contains both a and b as vertices on either side of x. Since a and b are not successors, there must be a neighbor vertex y to x in the interior of P. But then y is trapped since $\mathsf{dist}(y) \geq \mathsf{dist}(x) \geq \max_{v \in P} \mathsf{dist}(v)$. By Lemma 6, no vertex in G is trapped and so a and b not being successive is a contradiction.

Consider an order $n+1$ vertex a. For the sake of a contradiction, suppose that two of its children vertices x and y are not successive. We have shown that shell boundary $n+2$ is simple. Therefore, shell boundary $n+2$ and edges ax and ay form a triangle. Since x and y are not successive, there must be a vertex b within the interior of this triangle with $\mathsf{dist}(b) \leq \mathsf{dist}(a)$. However, this is impossible since this would leave b with no parent. □

[5]It can be shown that a non-disjoint set of shell boundaries can be made to be disjoint.

For a graph G satisfying Lemma 7, there exists a simple cyclic ordering of cousin vertices on each shell defined by the shell boundaries. For each shell, we simply follow the single shell boundary curve in a clockwise direction listing off the parents of the descendant edges in the order the edges are encountered. Vertices with multiple children will be listed multiple times, but since children occur successively, the final sequence will have all repeated vertices in succession. We therefore discard any sequence of the same vertex with a single vertex to obtain the final unique cyclic order. For a graph satisfying Lemma 7, we assume this cyclic order for the vertices on each shell and freely refer to successive cousins on a given shell. We will typically refer to vertices in a shell $\mathfrak{S}_n$ with labels $x_0^n, x_1^n, x_2^n, \ldots$ and, if there are q vertices in $\mathfrak{S}_n$, it is understood that $x_i^n \equiv x_j^n$ when $i \equiv j \pmod{q}$. The successor of x_i^n is x_{i+1}^n and the predecessor of x_i^n is x_{i-1}^n.

Lemma 8. *Suppose that $G \in \mathfrak{G}$ has the strong descendant property. Then cousin-edges occur only between successive cousins. A vertex has at most two cousin-edges and these edges are not successive.*

Proof. Given the properties of the embedding of G implied by Lemma 7, it is apparent that if two non-successive cousins share an edge, then at least one cousin is blocked from having any children. Therefore, only successive cousins can share an edge. Since cousin-edges exist only between successive cousins, and given the restriction of every vertex having a child and a parent, it is clear that if there are two cousin-edges, they do not appear successively. Rather, each is sandwiched directly between the set of children and the set of parent edges. □

Lemmas 6, 7, and 8 allow us to view a plane-embedded, VAP-free graph G with the strong descendant property in a convenient way. The simple boundary shells appear one inside the other. Between two shell boundaries n and $n+1$ lie the set of vertices in $\mathfrak{S}_n$. These vertices appear in a cyclic order[6] with each set of parent edges crossing directly through shell boundary n and each set of children edges crossing directly through shell boundary $n+1$. Cousin-edges appear directly between successive cousins. See Figure 3.3 for an example of our notation applied to the portion of a graph in $\mathfrak{G}$. This picture is of great use when visualizing various cases in the following theorems. When it is obvious from the theorem statement that the above lemmas hold, we might not refer to them explicitly.

Lemma 9. *Let $G \in \langle 3,6\rangle \cup \langle 4,4\rangle$. Then with respect to any fixed origin, all vertices have at most two parents and these parents must be successive on their shell. Furthermore, if a vertex x has two parents a, b such that a is the predecessor of b, then x is the rightmost child of a and the leftmost child of b.*

[6]Even if the original embedding makes it difficult to see their cyclic arrangement, they can be moved into a more visually cyclic pattern without changing the topology of the embedding.

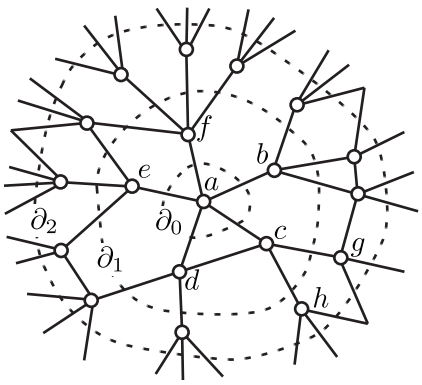

Figure 3.3: A portion of a plane-embedded, VAP-free graph with the strong descendant property. The origin vertex is labeled a and forms shell zero, denoted by $\mathfrak{S}_0$. The dashed lines denote the boundary curves, labeled ∂_0, ∂_1, and ∂_2. The first shell, $\mathfrak{S}_1$, consists of vertices b through f. Vertices c and d share a cousin edge. The rightmost child of c is h and the leftmost child of c is g. g and h are successors.

Proof. We use an inductive argument over shells of vertices and handle the cases of $\langle 3, 6\rangle$ and $\langle 4, 4\rangle$ simultaneously. In both cases, the origin has no parents, and the first level vertices only have the origin as their single parent. Therefore, the theorem is true for $\mathfrak{S}_0$ and $\mathfrak{S}_1$. We assume that the theorem is true for all shells up to and including $\mathfrak{S}_n$, and will to show that this implies it is true for $\mathfrak{S}_{n+1}$.

Let $x \in \mathfrak{S}_{n+1}$ and, for the sake of a contradiction, assume that x has three or more parents in $\mathfrak{S}_n$. By Lemma 7, the parents x occur successively and this means that there are three parents a, b, c of x such that a precedes b and b precedes c. If $G \in \langle 4, 4\rangle$, then b shares no cousin-edges since that would form a triangle. By the inductive hypothesis, b has at most two parents since $b \in \mathfrak{S}_n$. But, then the degree of b is at most three which contradicts $G \in \langle 4, 4\rangle$. Therefore, assume $G \in \langle 3, 6\rangle$. Now b can be incident on both a and c. Again, by the inductive hypothesis, b has at most two parents which implies that the total degree of b is at most five, a contradiction. The induction therefore goes through and we conclude that all vertices in G have at most two parents.

The final two claims do not require induction. For the penultimate claim, let $x \in \mathfrak{S}_{n+1}$ and suppose, for the sake of a contradiction, that x has non-successive parents $a, c \in \mathfrak{S}_n$. By the argument just given, these are the only parents of x. Therefore, for a and c to be non-successive, there must be at least two vertices $b, d \in \mathfrak{S}_n$ such that starting from a and traveling clockwise along boundary $n-1$, the order in which the four mentioned vertices are encountered is either a, b, c, d or a, d, c, b. It is then true that either the child of b or the child of d is trapped and this is impossible by Lemma 6.

For the final claim of the theorem, observe that if a shared child were not the rightmost child of the left parent, then the child of the left parent which succeeds the shared child would be trapped. By Lemma 6, no vertices are trapped. The

same holds true for the right parent. □

Theorem 10. *If $G \in \langle 4, 4\rangle$, then every vertex has at least two children, and the relationship between every non-origin vertex x and its non-children neighbors satisfies one of the following mutually exclusive cases:*

1. *x is incident on exactly one cousin-edge and x has exactly one parent.*
2. *x is not incident on any cousin-edge and x has exactly two parents.*
3. *x is not incident on any cousin-edge and x has exactly one parent.*

Finally, every non-origin shell has at least as many vertices as its predecessor shell.

Proof. By the definition of what it means to be on a particular shell (all vertices on a particular shell have the same shortest-path distance to the origin), it is trivially true that all non-origin vertices have at least one parent on a previous shell, and all its parents are on the previous shell.

We can proceed to prove the claim of lemma by induction on shell levels. The inductive hypothesis is that all of the claims of the lemma hold for the first through n-th shells.

The origin has at least four children and, these children comprise the set of vertices on the first shell. Since there are no triangles in the tessellation, there can be no cousin-edges on the first level. Therefore, all vertices on the first shell satisfy case 3, and each vertex has at least three children.

Now consider any vertex y on shell $n+1$. Since there are at least four unique vertices on the first shell, by the inductive hypothesis, we can conclude that there are at least four on the nth shell. For the sake of a contradiction, consider the situation where y has three parents. Call these parents a, b, c in clockwise order. By the inductive hypothesis, vertex b has at least two children. One of these children is y and now the other must be either a or c which is impossible since a and c are on the same level as b (furthermore, this would create a triangle). Therefore, all vertices on the $n+1$ shell level have at most two parents each. By the inductive hypothesis, all vertices on the nth shell have at least two children and therefore the level $n+1$ has at least as many vertices as level n.

According to the previous paragraph, there at least four vertices on level $n+1$. Let x and z be the distinct left and right shell-neighbors of vertex y. Furthermore, note that x and z are not shell-neighbors either since there is at least one additional vertex in between them.

For the sake of a contradiction, suppose y is incident on both x and z and let b be a parent of y. By the inductive hypotheses, b has an additional child. This child must be either x or z for if it were neither of them, then one of them would have no parents in the n-th shell. However, if x or z is also a child of b, then

this creates a triangle which is forbidden. Therefore, at most one of x and z is incident on y.

Suppose that y shares an edge with x and b is a parent of y. For the sake of a contradiction, suppose that y has an additional parent. It is easy to argue that this parent must be a or c. Suppose a is y's parent. Then x cannot be a child of a since this would form a triangle. The vertex immediately to the left of a must be x's parent, but this also leads to a contradiction since a would have only one child. Therefore only c can be the second parent of y. This too leads to a contradiction since b is blocked from having a second child. Therefore, if y shares an edge with a vertex on the same level, then it had exactly one parent.

Since we have shown that every vertex at level $n+1$ has at most two non-child neighbors, it follows that every vertex has at least two children. □

Corollary 11. *If $G \in \langle 4, 4+k \rangle$ where $k \geq 0$, then each vertex has at least $2+k$ children and there are at least a factor of k more vertices on each subsequent vertex shell.*

Proof. Since it remains true that every vertex on the $(n+1)$-st shell has at most 2 parents and since every vertex on the nth shell has at least $2+k$ children, there must be at least k distinct vertices on level $n+1$ for every vertex on level n. □

Theorem 12. *If $G \in \langle 3, 6 \rangle$, then every vertex has at least two children, at most two parents, and every non-origin shell has at least as many vertices as its predecessor shell.*

Proof. Observe that the origin has at least $q_{\min}$ children vertices which comprise the first shell. The vertices on the first shell each have only the origin as their parent. We can easily demonstrate that each vertex on the first shell can share an edge only with their immediate left and right shell-neighbors. For the sake of a contradiction, suppose that vertex x on the first shell shares an edge with vertex z two positions to the right of x. Let vertex y be the middle vertex. Vertex y can only connect to vertex x and vertex z which would give it a degree of at most three in violation of $q_{\min} \geq 6$. If each vertex on the first shell can share edges with at most their left and right shell-neighbors, then they each have at least three children, since $p_{\min} \geq 3$ and they have exactly one parent and are incident on at most two cousin-edges. Therefore, the lemma holds for the first shell.

Now assume that the lemma holds for the first through nth shells. Then shell n has at least $q_{\min}$ vertices and each of these has at least two children each. We first show that vertices on shell $n+1$ have at most two parents each. Let x, y, z be three consecutive neighbors on shell $n+1$ and a, b, c be three consecutive neighbors on shell n all occurring in clockwise order. For the sake of a contradiction, suppose that y has three parents on shell n, and suppose that these parents are vertices

a, b, and c. By the inductive hypothesis, b can have at most two neighbors and these neighbors must be a and c. But then b is forced to have at most one child and this is a contradiction. Therefore, each vertex on level $n+1$ has at most two parents. Just as in the base case of the induction we can now easily argue that each vertex on shell $n+1$ shares an edge with at most both of its shell-neighbors. Since every vertex has degree at least 6, each vertex on $n+1$ has at least two children. □

Corollary 13. *If $G \in \langle 3, 6+k \rangle$ where $k \geq 0$, then each vertex has at least $2+k$ children and there are at least a factor of k more vertices on each subsequent vertex shell.*

The proof follows easily, in a similar way to the proof of Corollary 11.

3.2 Combined Faults

We begin with some negative results which are easily obtained by constructive arguments.

Definition 6 (majority rule automaton)**.** The *majority-rule automaton* on a graph G is defined as follows. The medium $\mathbf{M} = V(G)$. For each a in $\mathbf{M}$, the inputs

$$U(a) = \begin{cases} N(a) & : \ |N(a)| \ \text{odd} \\ N(a) \cup \{a\} & : \ |N(a)| \ \text{even} \end{cases}$$

and the transition rule

$$\phi_a = \mathsf{maj}_{|U(a)|}$$

where maj_n denotes the majority function on n Boolean arguments.

Theorem 14. *Every majority-rule automaton on a graph in $[3,8] \cup [4,6] \cup [\infty, 4]$ is intolerant of combined faults.*

Proof. Figures 3.5, 3.6, and 3.4 illustrate the truth of each of the respective cases in this theorem. □

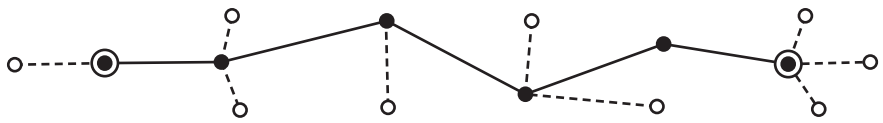

Figure 3.4: The circled black dots are manufacturing faults. The black dots constitute a self-sustaining island of error between the manufacturing fault dots.

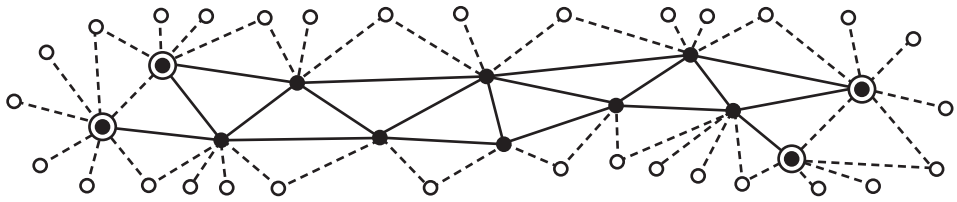

Figure 3.5: The circled black dots are manufacturing faults. The black dots constitute a self-sustaining island of error between the manufacturing fault dots.

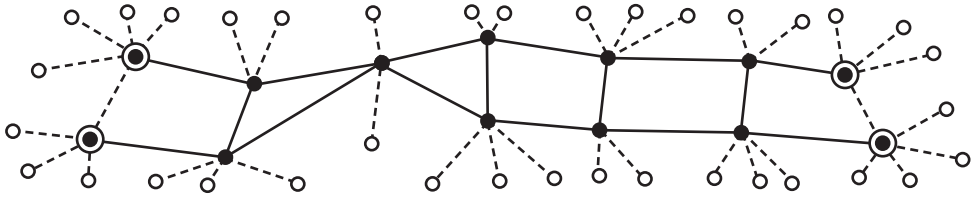

Figure 3.6: The circled black dots are manufacturing faults. The black dots constitute a self-sustaining island of error between the manufacturing fault dots.

For the remainder of this section, assume that all automata are monotone and binary, and are both zero- and one-preserving. This implies that every ϕ_a depends essentially[7] on at least one input. We are only interested in the stability of trajectories **0** and **1**. We claim that for monotone automata, there is a simple optimal strategy for the adversary that maximizes the probability of every point being in error. The strategy is the obvious *greedy strategy* where the adversary uses every fault to set the state of a point into error. Notice that this adversary requires no past, present or future knowledge of state of the cells in the automata or the allotment of faults. Therefore, when talking about monotone automata, the adversary needs much less power than the general omniscient adversary represented in Definition 1. The notion of the greedy adversary being optimal is encapsulated in the next Lemma. We will generally assume the use of this Lemma when talking about monotone binary automata and the **0** and **1** trajectories.

Lemma 15. *Let A be a monotone binary automaton and suppose that we are interested in the stability of the trajectories* **0** *and* **1**. *In this case, the greedy strategy is optimal.*

Proof. Let y be either the **0** or **1** trajectory. From Definition 1, the probability that a point a is in error is

$$(\mu_\alpha \times \nu_\beta)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}).$$

In a sense, the every possible "strategy" is represented in this definition. A strategy for the adversary is optimal if for every point $a \in \mathbf{L}$, every pair of fault

[7]An input x to a Boolean function is depended upon "essentially" if there is a setting of the other inputs such that the output of the function changes as the value of input x changes.

sets (F_M, F_T) included in the set in the above definition will also result in a being in error if the given strategy for the adversary is followed. Let (F_M, F_T) be a fault pair for which there exists a consistent[8] configuration x such that $a \in E_{x,y}$. Let $B \subseteq \mathbf{L} \cap F_C$ be the set of points where the greedy strategy was not followed. This means that all points in $B \notin E_{x,y}$. By definition, the greedy strategy would put all points $B \in E_{x,y}$. Since all functions are monotone, adding any points in B to $E_{x,y}$ cannot result in a not being in error. Clearly adding all points in B to $E_{x,y}$ results in a consistent configuration and furthermore, this configuration is exactly the greedy strategy. □

When dealing with monotone binary automata, it will often be easier to prove a result for a slightly weaker automaton. We mean "weaker" in the sense that it is strictly more likely to fail to remember a bit than the original. We formalize this notion in the following definition.

Definition 7 (weaker)**.** Let A and B be automata with a common medium and suppose that y is a trajectory in both automata. B is said to be *weaker* than A (with respect to trajectory y) if for every $a \in \mathbf{L}$, every fault set F_C that allows a to be in error in A, also allows a to be in error in B. By "allows a to be in error in A (or B)," we mean there exists a configuration x that is consistent with F_C and the transition rules of A (or B) that results in a being in error. If B is weaker then A, then we say that B is a *weakened version* of A.

In general it would be very difficult to prove that two automata satisfy this definition, but for monotone binary automata, there is one particular type of modification we can do for which it is obvious that the modified automaton is weaker than the original.

Definition 8 (pessimistic assumption)**.** Let A and B be monotone binary automata with identical media. For each $a \in \mathbf{M}$, let ϕ_a and ρ_a denote the transition functions associated with a and automata A and B respectively. Define ρ_a to be the same as ϕ_a except that we set some arbitrary subset of ρ_a's arguments to constant 1 if the desired trajectory $y = \mathbf{0}$, and 0 if $y = \mathbf{1}$. We call any such automaton B derived from A in this way a *pessimistic* version of A.

The idea of a pessimistic version is that cells are made to "pessimistically" believe that certain of their input cells are always in error even though these cells might not always be in error.

Lemma 16. *If A and B are monotone binary automata such that B is a pessimistic version of A. Then it follows that (1) B is weaker than A and (2) if B can remember a bit under the combined fault model (or variations such as the manufacturing or transient fault models), then A can also remember a bit under the same fault model.*

[8]We call a configuration x *consistent* if it satisfies $x \in C_y$ and $D_x \subseteq F_C$.

Proof. By the monotonicity of the transition functions, it trivially follows that a pessimistic version of an automaton is weaker. From the definition for stability under the combined fault model (Definition 1), and the special cases of stability under manufacturing and transient faults (Definitions 2 and 3), it is a weaker version of an automaton. □

In proving stability, we will routinely construct a pessimistic version of an automaton and prove stability on this "weaker" version of the automaton. For example, Theorem 17 is for automata that are weaker than the automata for which we are actually interested in proving properties. After this, we show how to weaken the automata in which we are actually interested by using a pessimistic construction. The conclusions for the weaker automata will then hold for the original automata using Lemma 16.

Definition 9 (dependency graphs)**.** The *medium dependency-graph* for an automaton A is denoted $\mathfrak{D}_{\mathbf{M}}(A)$. It has vertex set $V(\mathfrak{D}_{\mathbf{M}}(A)) = \mathbf{M}$, and edge set $E(\mathfrak{D}_{\mathbf{M}}(A)) = \{(a, b) : b \in U_{\mathbf{M}}(a)\}$. The *lattice dependency-graph* for an automaton A is denoted $\mathfrak{D}_{\mathbf{L}}(A)$. It has vertex set $V(\mathfrak{D}_{\mathbf{L}}(A)) = \mathbf{L}$, and edge set $E(\mathfrak{D}_{\mathbf{L}}(A)) = \{(a, b) : b \in U_{\mathbf{L}}(a)\}$.

A lattice dependency-graph is a directed, acyclic graph by definition. In general, a medium dependency-graph can be a directed multi-graph with cycles. Our strategy will be to modify (minimally) an automaton so as to avoid the existence of multiple paths between vertices in its dependency-graph—that is, we turn the graph into a forest.

Theorem 17. *Let A be a non-constant monotone binary automaton and let either $y = \mathbf{0}$ or $y = \mathbf{1}$ be the desired stable trajectory. For each $a \in \mathbf{M}$, let $h(a)$ represent the minimal number of inputs in error necessary to force ϕ_a into error. For all $a \in \mathbf{M}$, suppose $h(a) \geq 2$, and suppose there exists a real λ such that $|U(a)| \leq \lambda h(a)$. If the medium dependency-graph is a forest, then y is stable under the combined fault model.*

Proof. The significance of the medium dependency-graph being a forest is that for any finite set $A \subset \mathbf{L}_n$ of points at timestep n, $\Pr[a \in E_{x,y},\ \forall a \in A] = \prod_{a \in A} \Pr[a \in E_{x,y}]$. Now for any a, $\Pr[a \in E_{x,y}]$ is equal to the probability of the event that either $a \in F_C$ (a fault occurred at point a) or a sufficient number of a's inputs are in error to force a into error. Let $P_n = \sup_{a \in \mathbf{L}_n} \Pr[a \in E_{x,y}]$. Suppose that

$t(a) = n + 1$. Then

$$\begin{aligned}
\Pr[a \in E_{x,y}] &\leq \varepsilon + \sum_{\substack{A \subseteq U(a), \\ |A| \geq h(a)}} \Pr[b \in E_{x,y},\, \forall b \in A] \\
&\leq \varepsilon + \sum_{\substack{A \subseteq U(a), \\ |A| \geq h(a)}} {P_n}^{|A|} \;=\; \varepsilon + \sum_{k=h(a)}^{|U(a)|} \binom{|U(a)|}{k} {P_n}^k \\
&\leq \varepsilon + {P_n}^{h(a)} \sum_{k=1}^{\lambda h(a)} \binom{\lambda h(a)}{k} \\
&\leq \varepsilon + {P_n}^{h(a)} 2^{\lambda h(a)}.
\end{aligned}$$

Let $0 < \varepsilon \leq 1/2^{2\lambda+1}$. By Definition 1, $P_0 = \varepsilon$. Now suppose that $P_n \leq 2\varepsilon$. Noting that $h(a) \geq 2$ and $\lambda \geq 1$, we find

$$\Pr[a \in E_{x,y}] \leq \frac{1}{2^{2\lambda+1}} + \left(\frac{2}{2^{2\lambda+1}}\right)^{h(a)} 2^{\lambda h(a)} \leq 2\varepsilon.$$

Therefore $\sup_{a \in \mathbf{L}_{n+1}} \Pr[a \in E_{x,y}] = P_{n+1} \leq 2\varepsilon$ whenever $P_n \leq 2\varepsilon$. For all $a \in \mathbf{L}$, if the fault-rate is not greater than $1/2^{2\lambda+1}$, then $\Pr[a \in E_{x,y}] \leq 2\varepsilon$, and consequently $\lim_{\varepsilon \to 0} \sup_{a \in \mathbf{L}} \Pr[a \in E_{x,y}] = 0$. By definition, y is stable. □

Lemma 18. *Let $G \in \langle 5, 5\rangle$ and suppose that x is a vertex in G that has either a cousin-neighbor or two parents. Then x does not share its children with any cousin.*

Proof. The truth of this lemma is readily seen from the representative sections of a $\langle 5, 5\rangle$ graph shown in Figure 3.7. Note that the lemma fails if $p_{\min}$ is lowered to 4. □

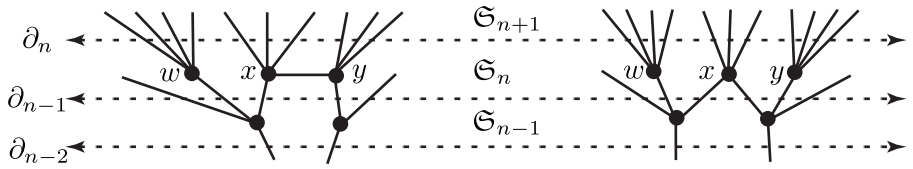

Figure 3.7: Two possible sections of a $\langle 5, 5\rangle$ graph. Vertices w, x, and y are on shell $\mathfrak{S}_n$. The dashed lines denote the n- through $(n-2)$-th boundary curves denoted ∂_n, ∂_{n-1}, and ∂_{n-2}.

Theorem 19. *Every majority-rule automaton on a graph in $\langle 3, 13\rangle \cup \langle 4, 9\rangle \cup \langle 5, 7\rangle$ can remember a bit with combined faults.*

Proof. Let B and A be automata with transition functions denoted by ρ and ϕ respectively. We define B to be a pessimistic version of A where A is a majority-rule automaton on a graph in $\langle 3, 13\rangle \cup \langle 4, 9\rangle \cup \langle 5, 7\rangle$. The transition functions in A are majority functions which are just cases of threshold functions. The thresholds for ϕ_a and ρ_a denote the minimum numbers of inputs sufficient to force a into error. For ϕ_a, this is always $\lfloor (d(a)+1)/2 \rfloor$. If we assume that k of ϕ_a's inputs are in error, then this has the effect of lowering the threshold by k. In the remainder of the proof and the next, we will describe ρ_a by describing which inputs "we assume to be in error" or equivalently, which inputs "we ignore."

For each $a \in \mathbf{M}$, we ignore the inputs from its cousins, parents, leftmost child, and if $d(a)$ is even, a itself. By Lemma 9, each vertex has a unique leftmost child and the leftmost child and the rightmost child are the only children vertices a might share with a cousin. By Lemma 8, if a does share a child, it is only with a successive cousin.

We claim that B's medium dependency-graph $\mathfrak{D}_{\mathbf{M}}(B)$ is a forest. Clearly, all paths in $\mathfrak{D}_{\mathbf{M}}(B)$ are strictly increasing in shell level as the path is traversed. Furthermore, consider distinct vertices a and b on shell n. By Lemma 9, the only child a and b can share is a child which is a leftmost child for one vertex and a rightmost child for the other vertex, but all leftmost child edges have been deleted; therefore, a and b have no common child on the next child. By induction on shell levels, no paths starting from a and b can ever merge. Thus, $\mathfrak{D}_{\mathbf{M}}(B)$ is a forest.

In what follows, let us define $c(a)$ to be the number of input-cells on which a cell a essentially depends. We define $h(a)$ to be the minimum number of input-cells necessary to ensure a is in error.

If $A \in \langle 3, 13\rangle$, by Lemma 8 and Theorem 12 we might ignore up to 5 neighbors for odd degree vertices and 6 neighbors for even degree vertices. By the strong descendant property, every vertex has at least one parent and one child, so we ignore at least 2 neighbors. Suppose $d(a)$ is odd, then

$$\frac{d_A(a)+1}{2} - 5 \leq h(a) \leq \frac{d_A(a)+1}{2} - 2, \quad d_A(a) - 5 \leq c(a) \leq d_A(a) - 2, \quad 13 \leq d_A(a),$$

from which we conclude

$$c(a) \leq 2h(a) + 7, \quad 2 \leq h(a), \quad 8 \leq c(a), \quad h(a) \leq c(a).$$

If $d_A(a)$ is even, then

$$\frac{d_A(a)+2}{2} - 6 \leq h(a) \leq \frac{d_A(a)+2}{2} - 2, \quad d_A(a) - 6 \leq c(a) \leq d_A(a) - 2, \quad 14 \leq d_A(a),$$

from which we conclude

$$c(a) \leq 2h(a) + 8, \quad 2 \leq h(a), \quad 9 \leq c(a), \quad h(a) \leq c(a).$$

If $A \in \langle 4, 9\rangle$, by Theorem 10 we might ignore up to 3 neighbors for odd degree vertices and 4 neighbors for even degree vertices. By the strong descendant property, every vertex has at least one parent and one child, so we ignore at least 2 neighbors. Suppose $d(a)$ is odd, then

$$\frac{d_A(a)+1}{2}-3 \leq h(a) \leq \frac{d_A(a)+1}{2}-2, \quad d_A(a)-3 \leq c(a) \leq d_A(a)-2, \quad 9 \leq d_A(a),$$

from which we conclude

$$c(a) \leq 2h(a)+3, \quad 2 \leq h(a), \quad 6 \leq c(a), \quad h(a) \leq c(a).$$

If $d_A(a)$ is even, then

$$\frac{d_A(a)+2}{2}-4 \leq h(a) \leq \frac{d_A(a)+2}{2}-2, \quad d_A(a)-4 \leq c(a) \leq d_A(a)-2, \quad 10 \leq d_A(a),$$

from which we conclude

$$c(a) \leq 2h(a)+4, \quad 2 \leq h(a), \quad 7 \leq c(a), \quad h(a) \leq c(a).$$

To handle the case of $A \in \langle 5, 7\rangle$, we will need to take a slightly more delicate approach. As before, we ignore all cousin-neighbors, parents, and a itself if $d(a)$ is even. By Theorem 10, we can classify vertices into three types: (1) a is incident on exactly one cousin-edge and a has exactly one parent, (2) a is not incident on any cousin-edge and a has exactly two parents, (3) a is not incident on any cousin-edge and a has exactly one parent. For type (1) and (2) vertices, we do not ignore any children. For type (3) vertices, we ignore just the leftmost child as before.

By Lemma 18, a type (1) or (2) vertex does not share its children with any cousins. Therefore, the only type of vertices that can share their children are type (3) and this sharing is eliminated as before by ignoring their leftmost children. The medium dependency-graph for B is therefore a directed tree and we might ignore up to 2 neighbors for odd degree vertices and 3 neighbors for even degree vertices. By the strong descendant property, every vertex has at least one parent and one child, so we ignore at least 2 neighbors. Suppose $d(a)$ is odd, then

$$h(a) = \frac{d_A(a)+1}{2} - 2, \quad c(a) = d_A(a) - 2, \quad 7 \leq d_A(a),$$

from which we conclude

$$c(a) = 2h(a)+1, \quad 2 \leq h(a), \quad 5 \leq c(a), \quad h(a) \leq c(a).$$

If $d_A(a)$ is even, then

$$h(a) = \frac{d_A(a)+2}{2} - 3, \quad c(a) = d_A(a) - 2, \quad 8 \leq d_A(a),$$

from which we conclude

$$c(a) \leq 2h(a) + 2, \quad 2 \leq h(a), \quad 6 \leq c(a), \quad h(a) \leq c(a).$$

Therefore, for B derived from $A \in \langle 3, 13\rangle \cup \langle 4, 9\rangle \cup \langle 5, 7\rangle$, it is true that $|U(a)| \leq \lambda h(a)$ where $\lambda = 6$. Therefore B satisfies all the conditions of Theorem 17 and can therefore tolerate combined failures. Since B is a pessimistic version of A, by Lemma 16, A is also tolerant of combined faults. □

Theorems 19 and 14 together provide a natural set of bounds for positive and negative results concerning the combined fault model. The following theorem almost entirely fills in the gap between Theorems 19 and 14 except for the additional restriction of bounded cell input degrees. We conjecture that the bounded input degree restriction is unnecessary. The intuition for this conjecture is that for positive results, larger cell-input neighborhoods only seem to help. We only require the restriction due to our use of Lemma 22 which uses the restriction to enumerate the number of excuse graphs of a certain size.

Theorem 20. *Every majority-rule automaton on a graph in* $(\langle 3, 9\rangle \cup \langle 4, 7\rangle \cup \langle 5, 5\rangle) \cap \mathcal{Q}_{\text{bdd}}$ *is tolerant of combined faults.*[9]

We delay the proof of Theorem 20 until after some lemmas and definitions have been given. At a high level, our proof strategy roughly follows that of [20] (Appendix A). Given a point $r \in \mathbf{L}$, which we refer to as "the origin," we want to bound the probability that r is in the error set. To do this, we associate with each configuration x, trajectory y, and combined fault set F_C, a graph $\widetilde{G}(r, F_C, x, y)$. We call $\widetilde{G}(r, F_C, x, y)$ a *flat excuse graph*. As usual, we assume that y is either $\mathbf{0}$ or $\mathbf{1}$ and that the adversary employs the optimal strategy, the greedy strategy, that maximizes the probability of r (and all other points) being in error. Strictly speaking, it is redundant to give both the fault set F_C and the configuration x since x is determined from F_C and y when the greedy strategy is assumed. For a given r and y, we denote the set of all possible $\widetilde{G}(r, F_C, x, y)$ as $\widetilde{\mathcal{G}}(r, y)$. We show that each graph $G \in \widetilde{\mathcal{G}}(r, y)$ has a probability of occurrence at most $\varepsilon^{|E(G)|/27}$. We also show that the number of such graphs with k edges is at most $(2q_{\max})^{2k}$ where $q_{\max}$ is the upper-bound on cell input degrees for the automaton. Using the union bound, we get an upper-bound on $\Pr[r \in E_{x,y}]$ in terms of ε.

As was the case previously, it is simpler to work with automata that have acyclic medium dependency-graphs. In Theorem 19, we were able to reduce the automata so that their medium dependency-graphs were forests, but here we have to settle for acyclic. To achieve acyclic dependency graphs, we construct weakened versions of the automata. The construction is centered around an arbitrarily chosen cell, $a^* \in \mathbf{M}$, that we call the *center-cell*. For a majority automaton A on a graph in $\langle 3, 9\rangle \cup \langle 4, 7\rangle \cup \langle 5, 5\rangle$ with center-cell a^*, we denote the weakened version of the automata $W(A, a^*)$ according to the construction described below.

[9] See page 20 for definition of $\mathcal{Q}_{\text{bdd}}$.

Begin by assuming that $W(A, a^*)$ is a copy of A. As was done in the proof of Theorem 19, we describe which inputs each cell "ignores" (or equivalently, "assumes to be in error"). With a^* as the the "root," terms such as "parent," "child," and "cousin-neighbor" are well-defined and we continue to use these terms as they were used previously. In all cases, if a cell depends on itself in automaton A, then assume that the cell ignores itself in $W(A, a^*)$

When A is a majority automaton on a graph in $\langle 3, 9\rangle$, automaton $W(A, a^*)$ has the following modifications for each cell: (1) all parent inputs are ignored, (2) if the cell does not have two parents and two cousin inputs, then all cousin inputs are ignored.

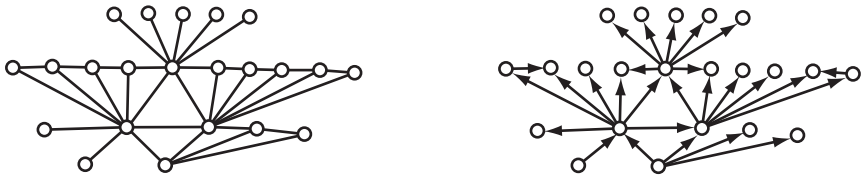

Figure 3.8: A section of an automaton A on $\langle 3, 9\rangle$ modified to $W(A, a^*)$.

When A is a majority automaton on a graph in $\langle 4, 7\rangle$, automaton $W(A, a^*)$ has the following modifications for each cell: all parent and cousin inputs are ignored. Note that by ignoring *all* cousin inputs, the resulting graph is a *ranked graph*. A ranked (acyclic) graph has the property that all walks between a given pair of vertices are of the same length. Although we could use the ranked-graph property to simplify the proofs for cases based on $\langle 4, 7\rangle$, we choose not to use this property and let $\langle 4, 7\rangle$ be handled within a more general framework.

When A is a majority automaton on a graph in $\langle 5, 5\rangle$, we describe automaton $W(A, a^*)$ by modifying vertices shell-by-shell, starting from the center-cell a^*. The modifications to any shell n depend on the modifications to the previous shell $n - 1$. The center-cell (shell zero) is left unchanged. Assuming shell $n - 1$ has already been modified, shell n receives the following modifications. For each cell on shell n: (1) all parent inputs are ignored, (2) cousin inputs are ignored except for when the threshold of its parent (on modified shell $n - 1$) is less than two.

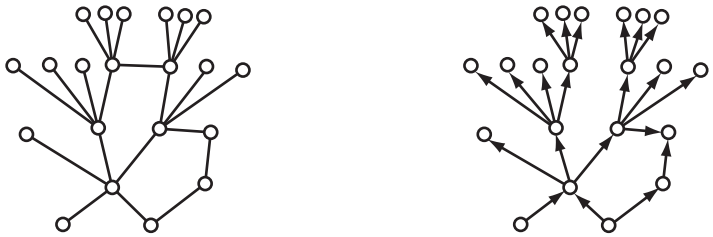

Figure 3.9: A section of an automaton A on $\langle 5, 5\rangle$ modified to $W(A, a^*)$.

Automaton $W(A, a^*)$ and its medium dependency-graph $\mathfrak{D}_{\mathbf{M}}(W(A, a^*))$ have several properties that we summarize in the following Lemma.

Lemma 21. *Let $W(A, a^*)$ be a modified automaton for some majority-rule automaton A on a graph in $\langle 3, 9\rangle \cup \langle 4, 7\rangle \cup \langle 5, 5\rangle$ with center-cell a^*. Then $W(A, a^*)$ has the following properties:*

1. *All walks on $\mathfrak{D}_{\mathbf{M}}(W(A, a^*))$ increase by one shell level at least every second step of the walk (and are therefore acyclic).*

2. *Every cell has a threshold of at least one.*

3. *No two consecutive cells along a walk have thresholds less than two.*

Proof. With the construction of $W(A, a^*)$ described above, and with the properties previously shown for graphs in $\langle 3, 9\rangle \cup \langle 4, 7\rangle \cup \langle 5, 5\rangle$, proving the above properties is largely routine. The proof of each property when A is a graph on $\langle 5, 5\rangle$ is the most difficult and so we sketch these proofs in some detail. We do not provide the proofs for A on $\langle 3, 9\rangle \cup \langle 4, 7\rangle$ since their proofs are similar, but easier. Therefore, assume below that A is an automaton on a graph in $\langle 5, 5\rangle$.

Theorem 10 classifies cells in $\langle 5, 5\rangle$ into three types: (type I) the cell has exactly one cousin-neighbor and exactly one parent, (type II) the cell has no cousin-neighbors and exactly two parents, and (type III) the cell has no cousin-neighbors and exactly one parent.

We begin by proving property (1). Eliminating child-to-parent edges (parent inputs) ensures that all walks are strictly non-decreasing in shell levels. By Theorem 10, all vertices have at most one cousin-neighbor. There are a number of other useful properties that hold for graphs in $\langle 5, 5\rangle$:

- A type III vertex always occurs successively with another type III vertex (on the same shell).

- A pair of successive type I vertices are always succeeded and preceded by a type III vertex.

- A type II vertex is always succeeded and preceded by a type III vertex.

- There is no edge between type II vertices.

- If there is a cousin-edge between vertices a and b, then at most one of the parents of a and b can be of type I or II.

Each of the above properties can be shown to hold with some case analysis similar to that used in the proof of Lemma 18. Taken together, the above properties allow us to prove the remainder of property (1).

Property (2) follows trivially from the fact that every cell can ignore at most two input-cells, and the threshold to begin with is at least three.

To prove property (3), assume, with an eye to obtaining a contradiction, that there exists a cell a having threshold one whose input-cell b also has threshold one. All type III cells have thresholds greater than one which implies a and b are each of type I or II. We also note that a and b are all either rightmost children, leftmost children, or cousin-neighbors of their parents. This is true because all other input-cells of parents are of type II. Suppose that a is of type II. It can be easily shown that no type II cell can have a type II as an input (this does not hold for graphs in $\langle 4, 7\rangle$ in general). This implies that b is of type I. By our modification, b does not ignore its cousin-neighbor since b's parent has threshold one. Therefore, b has threshold greater than one, and this leads to a contradiction. Consequently, vertex a must be of type I and the parent of a must be of type III. But then b must be of type I as well since the parents of any type II cell must both be of type I. By our modification, b does not ignore its cousin-neighbor since its parent a did ignore its cousin-neighbor. Therefore, the threshold of b is greater than one, and this is also a contradiction. □

The choice of center-cell a^* in the definition of $W(A, a^*)$ is arbitrary because all properties we are interested in hold equally well for all choices of center-cell. For a given automaton $W(A, a^*)$ with desired trajectory $y \in \{\mathbf{0}, \mathbf{1}\}$, we can now define the flat excuse graph $\widetilde{G}(r, F_C, x, y)$ mentioned earlier. We describe $\widetilde{G}(r, F_C, x, y)$ constructively with a pseudo-code algorithm that is a modified version of breadth-first-search (the pseudo-code style is borrowed from [5]). The algorithm produces the flat excuse graph $(N \cup T, E)$ where sets N and T are a partition of the vertices (which are points) called *non-terminals* and *terminals*, respectively.

```
CREATE-G̃(r, F_C, x, y)
    N ← ∅, T ← ∅, E ← ∅, Q ← NIL                    ▷ Initialization
    if r ∈ E_{x,y} ∧ r ∈ F_C
       then T ← T ∪ {r}
    if r ∈ E_{x,y} ∧ r ∉ F_C
       then N ← N ∪ {r}
            ENQUEUE(Q, r)
    while Q ≠ NIL
          do a ← DEQUEUE(Q)
             for each b ∈ U(a) ∩ E_{x,y}
                  do if ∃c ∈ N ∪ T s.t. c_ = b_      ▷ check if b_ visited already
                        then E ← E ∪ {(a, c)}
                        else  E ← E ∪ {(a, b)}
                              if b ∈ F_C
                                 then T ← T ∪ {b}
                                 else  N ← N ∪ {b}
                                       ENQUEUE(Q, b)
    return (N ∪ T, E)
```

The flat excuse graph $(N \cup T, E)$ is a directed graph. Excluding edges added with the line "**then** $E \leftarrow E \cup \{(a, c)\}$," the flat excuse graph is a finite subgraph of the lattice dependency-graph for $W(A, a^*)$. The edges added with line "**then** $E \leftarrow E \cup \{(a, c)\}$" are useful for accounting purposes and we will use them when estimating the number of possible flat excuse graphs, and when estimating the ratio of terminal to non-terminal vertices. From the algorithm, it is clear that every vertex in the graph corresponds to a distinct cell in the medium: all vertices in $N \cup T$ come from the queue, and there is a check to ensure that only vertices corresponding to a cell not already in the $N \cup T$ are added to the queue. The algorithm eventually terminates, and hence produces a finite graph, because there are only a finite number of points reachable from any point in the automaton's lattice dependency-graph. The decision not to enqueue points corresponding to cells previously enqueued represents the (conservative) decision to lower the threshold of a cell's transition function for a point whose corresponding cell has an input neighbor cell with a corresponding point already in the flat excuse graph. An important property of a flat excuse graph is that its probability of existence is bounded above by the fault-rate ε raised to a power equal to the number of terminals T. The probability of the event where the origin is in error is bounded by the probability of the event that the algorithm produces a non-empty excuse graph.

The following two figures provide an example of the construction of a flat excuse graph. The left-hand diagram in Figure 3.10 shows a finite section of the medium dependency graph for an automaton while the right-hand diagram in the same figure shows a finite section of the corresponding lattice dependency graph. Assume that the error-threshold for each cell is two. The leftmost diagram in Figure 3.11 shows an example fault set for the lattice as well as the resulting error set. In the middle and rightmost diagrams, we illustrate the resulting flat excuse graph. Notice that edge (b_{-1}, c_{-1}) is added while edge (b_{-1}, c_{-2}) is ignored (as illustrated with a dashed edge). The edge (b_{-1}, c_{-1}) is a pseudo-edge because b_{-1} does not actually depend on c_{-1}. We include the pseudo-edge in the final flat excuse graph to indicate that the threshold for the originating point (b_{-1} in this case) is lowered by one. The resulting flat excuse graph is shown in the rightmost diagram with the relative timestep indicators removed. The timesteps can be reconstructed by finding the length of a shortest path from a (the origin) to the point in question. Because of this, we can actually ignore time and view flat excuse graphs as being a graph on the medium rather than the lattice.

Lemma 22. *Let $W(A, a^*)$ be a weakened automaton for some majority automaton A on a graph in $(\langle 3, 9\rangle \cup \langle 4, 7\rangle \cup \langle 5, 5\rangle) \cap \mathcal{Q}_{\text{bdd}}$. Let $q_{\max}$ be the upper-bound on cell input degrees (according to restriction $\mathcal{Q}_{\text{bdd}}$). Then for every $k \geq 0$, $a \in \mathbf{M}$, $r \in \mathbf{L}$, and trajectory $y \in \{\mathbf{0}, \mathbf{1}\}$,*

$$|\{G \in \widetilde{\mathcal{G}}(r, y) : |E(G)| = k\}| \leq (2q_{\max})^{2k}.$$

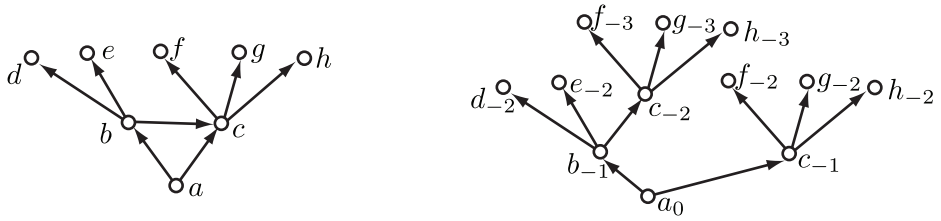

Figure 3.10: The left diagram is a section of a medium dependency graph for an example automaton, and the right diagram is the corresponding section of its lattice dependency graph. The origin is a_0, and the subscripts on points indicate their timestep difference relative to the origin (which we have given timestep zero).

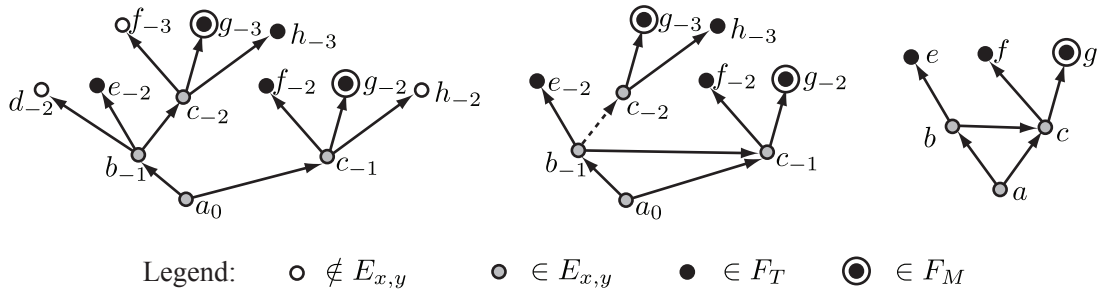

Figure 3.11: This series of diagrams illustrates the creation of a flat excuse graph for the example automaton in the previous figure. See the preceding text for a detailed discussion of the meaning of the diagram sequence.

Proof. We bound the number of flat excuse graphs having k edges by estimating the number of Euler circuits of length $2k$ consistent with our construction of the flat excuse graph. Every undirected graph has an Euler circuit that traverses each edge precisely twice (just double each edge and then every vertex has even degree). We are given that the degree of every vertex is at most $q_{\max}$. Consider the construction of an Euler circuit starting from the origin r. At each vertex along the circuit, there are at most $q_{\max}$ edges to choose from with at most two orientations each. The inequality in the theorem statement immediately follows. □

Lemma 23. *Let $W(A, a^*)$ be a weakened automaton for some majority automaton A on a graph in $\langle 3, 9\rangle \cup \langle 4, 7\rangle \cup \langle 5, 5\rangle$. Then for every $a \in \mathbf{M}$, $r \in \mathbf{L}$, and trajectory $y \in \{\mathbf{0}, \mathbf{1}\}$, if $G \in \widetilde{\mathcal{G}}(r, y)$ then*

$$27\,|V(G) \cap F_C| \geq |E(G)|\,.$$

Proof. Suppose that A is an automaton on a graph from $\langle 5, 5\rangle$, and let G be a flat excuse graph as stated in the lemma. By the definition of a flat excuse graph, the

vertex set is partitioned into a set of non-terminals N, and a set of terminals T. By definition, G is a directed, acyclic, and connected graph rooted at point r (not to be confused with the center-cell a^*). The non-terminal vertices have non-zero out-degree and terminal vertices have zero out-degree. Only r has in-degree zero. Our strategy is to show a fixed-fraction gap between the average out-degree of non-terminals and the overall average in-degree. Such a gap proves that there is a fixed fraction of terminal vertices (since the overall average out-degree is equal to the overall average in-degree in any graph).

(Proof that average out-degree of non-terminals is $\geq 3/2$.) The following properties hold for all $a \in N$:

- If $a = r$, then a has out-degree two or greater.
- Vertex a has at most one child with out-degree one.
- If a has out-degree one, then all of its children have out-degree two or greater.
- Any two vertices a and b with out-degree one do not share a neighbor c with out-degree two. Proof: we can quickly conclude that a must be of type I (that is, c has two parents, not a cousin-neighbor and a parent) since if b shared a cousin-edge with c, b would only have a directed edge to c because it needed out-degree two, a contradiction. As a general property of $\langle 5, 5\rangle$ graphs, both of the parents of a type I vertex are type III vertices, and type III vertices have out-degree two by the construction of $W(A, a^*)$.

Therefore, simple accounting says that there are at least as many vertices with out-degree two or more as there are vertices with out-degree one. Thus, the average out-degree of non-terminal vertices is at least 3/2.

(Proof that average in-degree of all vertices is $\leq 4/3$.) Except for r, all vertices have in-degree at least one but not more than two. Any type-II vertex has in-degree two and has no input-cells with in-degree two (by Theorem 10 and our modification). The remaining vertices are of types I or III and have in-degree one (by Theorem 10) and at most one of their children is of type II. We can easily show that all children of a vertex with in-degree two have in-degree one, and any vertex with in-degree one has at most one child with in-degree two. This, combined with the fact that every second non-terminal vertex has at least two input-cells (because their threshold is at least two), lets us conclude that the average in-degree is at most 4/3.

In any graph the total in-degree equals the total out-degree (which is also the number of edges in the graph). We therefore have the inequality

$$\frac{4}{3}\left(|N| + |T|\right) \geq \frac{3}{2}\,|N| \iff 8\,|T| \geq |N|\,.$$

The set of terminals T is equivalent to $V(G) \cap F_C$. Graph G is planar, and this lets us use Euler's Formula (see [7, Corollary 4.2.10]) to conclude $|V(G)| \geq |E(G)|/3$. Since $|N| + |T| = |V(G)|$, we have

$$27\,|V(G) \cap F_C| \geq |E(G)|.$$

The analysis for $\langle 3, 9\rangle \cup \langle 4, 7\rangle$ is similar to the analysis for $\langle 5, 5\rangle$, but is simpler. We therefore omit the details of the proof and just give the bounds.

For the case when A is on a graph in $\langle 4, 7\rangle$, by the construction of $W(A, a^*)$, it is immediately true that the out-degree of every non-terminal is at least 2. With some simple case analysis, the average in-degree of all vertices can be shown to be at most $3/2$. By the same argument as above for $\langle 5, 5\rangle$, this gives the inequality

$$12\,|V(G) \cap F_C| \geq |E(G)|.$$

For the case when A is on a graph in $\langle 3, 9\rangle$, by the construction of $W(A, a^*)$, it is immediately true that the out-degree of every non-terminal is at least 2. With some simple case analysis (but slightly more complicated than is required for $\langle 4, 7\rangle$), the average in-degree of all vertices can be shown to be at most $7/4$. This gives the inequality

$$24\,|V(G) \cap F_C| \geq |E(G)|.$$

Therefore, the inequality $27\,|V(G) \cap F_C| \geq |E(G)|$ holds for all cases. □

From Lemmas 22 and 23, the proof for Theorem 20 easily follows.

Proof (of Theorem 20). Let A be an automaton on a graph in $(\langle 3, 9\rangle \cup \langle 4, 7\rangle \cup \langle 5, 5\rangle) \cap \mathcal{Q}_{\text{bdd}}$. Let $q_{\max}$ be the maximum cell input degree according to restriction $\mathcal{Q}_{\text{bdd}}$. Arbitrarily choose a center-cell $a^* \in \mathbf{M}$. Now let $B = W(A, a^*)$ be a weaker version of A according to the description for $W(A, a^*)$ given above. By Lemma 16, if both trajectories $y = \mathbf{0}$ and $y = \mathbf{1}$ are stable for B, then both trajectories are also stable for A—in other words, if B can remember a bit, then so can A. Therefore assume B is the automaton with y as the desired stable trajectory. Let r be an arbitrary point. Then by Lemmas 22 and 23:

$$\begin{aligned}
\Pr[r \in E_{x,y}] &\leq \sum_{k \geq 0} \sum_{\substack{G \in \widetilde{\mathcal{G}}(r,y), \\ |E(G)| = k}} \Pr[G] \\
&\leq \sum_{k \geq 0} 2{q_{\max}}^{2k} \varepsilon^{k/27}
\end{aligned}$$

Therefore, $\lim_{\varepsilon \to 0} \sup_{r \in \mathbf{L}} \Pr[r \in E_{x,y}] = 0$ and y is stable for B by definition. □

3.3 Transient Faults

In the this section, we fill in some of the gap that remains in the classification of majority-rule automata on various tessellations under the transient fault model. Our next result (Theorem 24) shows that any majority-rule automaton on graphs in $[3,6] \cup ([\infty,4] \cap \mathcal{P}_{\text{fnt}}) \cup [\infty,2]$ are intolerant of transient faults. Theorem 20 says that all automata on graphs in $(\langle 3,9\rangle \cup \langle 4,7\rangle \cup \langle 5,5\rangle) \cap \mathcal{Q}_{\text{bdd}}$ are tolerant of combined faults, while Theorem 14 says that automata on graphs in $[3,8] \cup [4,6] \cup [\infty,4]$ are intolerant of combined faults. Ignoring the restriction of $\mathcal{Q}_{\text{bdd}}$ (which can be dropped for automata on graphs in $\langle 3,13\rangle \cup \langle 4,9\rangle \cup \langle 5,7\rangle$ according to Theorem 19), this leaves open the question of transient-fault tolerance for automata on graphs in $\langle 3,7\rangle \cup \langle 4,5\rangle \cup \langle \infty,3\rangle$. With some natural restrictions (which are mostly technical and which we conjecture to be unnecessary), we show that automata in this class are tolerant of transient faults.

Theorem 24. *Every majority-rule automaton on a graph in* $[3,6] \cup ([\infty,4] \cap \mathcal{P}_{\text{fnt}}) \cup [\infty,2]$ *is intolerant of transient faults.*

Proof. In all cases, we demonstrate the existence of a finite self-sustaining island of errors that includes an arbitrary cell a. Let $\mathbf{0}$ or $\mathbf{1}$ be the desired stable trajectory. We ignore the trivial case where a is in a finite component of the graph.

Suppose that A is an automaton on a graph in $\in [3,6]$. Consider the set of cells $U_{\mathbf{M}}(a)$. By the restrictions on cells in $U_{\mathbf{M}}(a)$, every cell in $U_{\mathbf{M}}(a)$ receives a majority of its inputs from other cells in $U_{\mathbf{M}}(a)$. Therefore, $U_{\mathbf{M}}(a)$ forms a self-sustaining island of error when a cell in $U_{\mathbf{M}}(a)$ is simultaneously in error.

Suppose that A is an automaton on a graph in $\in [\infty,4] \cap \mathcal{P}_{\text{fnt}}$. Then a is a vertex on finite cycle C in the graph. Suppose that the cells at every vertex on C are simultaneously in error. By the degree restrictions on cells in C, every cell on C receives a majority of its inputs from other cells on C. Therefore, cycle C forms a self-sustaining island of error when every cell in C is simultaneously in error.

Finally, suppose that A is an automaton on a graph in $[\infty,2]$ and let b be a neighbor of a. If a and b are ever both simultaneously in error, then they remain in error on all subsequent steps regardless of whether a and b also have neighbor cells. Therefore a and b in error form a self-sustaining finite island of error. □

The next theorem (Theorem 25) shows that there exist automata on graphs in $\langle \infty,3\rangle$ that are tolerant of transient errors, and therefore the restriction of $[\infty,4]$ to $\mathcal{P}_{\text{fnt}}$ is necessary. With the proof of Theorem 25 (and all other results in this chapter), as a corollary, we will have given the complete classification of majority-rule automata on the regular tessellation graphs $\{p,q\}$ for both the transient and the combined fault models. These results for the regular tessellations, which fall out as special cases from our more general tessellations, are summarized in Figure 3.12

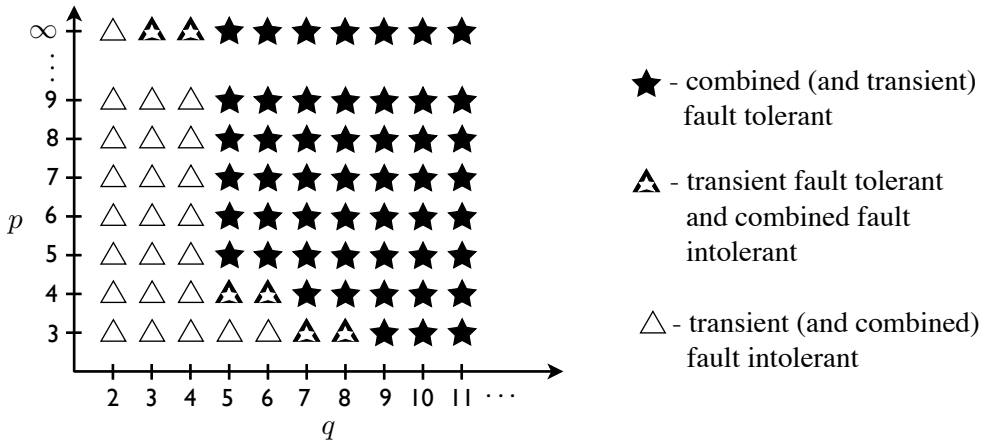

Figure 3.12: A summary of results for majority-rule automata on the regular tessellations $\{p, q\}$.

Theorem 25. *Every majority-rule automaton on a graph in* $((\langle 3, 7\rangle \cap \mathcal{P}_{\text{bdd,par}}) \cup (\langle 4, 5\rangle \cap \mathcal{P}_{\text{fnt,par}}) \cup \langle \infty, 3\rangle) \cap \mathcal{Q}_{\text{bdd}}$ *can remember a bit under the transient fault model.*

Before presenting the proof of Theorem 25, we present some supporting definitions and results. The key result on which Theorem 25 relies is the theorem of Toom [30] to which we referred in the Introduction as "Toom's Theorem." We state the theorem almost exactly as it appears in [30], and we indicate where our terminology differs in the footnotes.

Theorem 26 (Theorem 1 of Toom in [30])**.** *An automaton*[10] *and trajectory*[11] y *are given. Let there be* n *real functions* $L_1(\cdot), \ldots, L_n(\cdot)$ *on* $\mathbf{L}$ *and two numbers* $r > 0$, $R > 0$, *so that the following four conditions hold for all points* a, b, *all* k *from* 1 *to* n, *and all* $x_{U(a)} \in X_{U(a)}$*:*

1. $|U(a)| \leq R$; $|\{c : a \in U(c)\}| \leq R$.
2. $b \in U(a) \implies |L_k(b) - L_k(a)| \leq 1$.
3. $\sum_{k=1}^{n} L_k(a) = 0$.
4. $\phi_a(x_{U(a)}) \neq y_a \implies \exists c \in U(a)$ *such that* $x_c \neq y_c$ *and* $L_k(c) - L_k(a) \geq r$.

Then y *is a stable*[12] *trajectory of the automaton.*

[10] Toom calls an automaton a "combine."

[11] Toom lets y be an arbitrary configuration, but then points out that only trajectories can satisfy the theorem.

[12] Toom defines stability by taking a supremum over a set of measures. Our definition of stability can be shown to be equivalent to Toom's.

Toom's Theorem is a generalization of the technique first used by Toom to show that "Toom's Rule" is tolerant of transient faults. The proof involves a fairly technical construction, and we refer the interested reader to the paper where it appears for details. Although it is a very general theorem, it seems to have only been used (by Toom and others) to prove results for automata with media that are in finite-dimensional Euclidean spaces. The proof of Theorem 25 is likely its first application to automata on graphs that require hyperbolic space to be embedded with bounded distortion.

Toom's Theorem requires a way of assigning addresses to cells. For cells on a medium indexed by $\mathbf{Z}^d$, the addressing scheme is natural and trivial. For a regular hyperbolic tessellation, there are likely natural addressing schemes based on their finite group presentation, but even then, it is not obvious that such addressing schemes are well suited to the needs of Toom's Theorem. Furthermore, we require addressing schemes for relatively unstructured hyperbolic tessellations that satisfy certain global properties. Our strategy is specialized to the requirement of easily determining (close-to) shortest-path distances between vertices. The shortest-path-distance requirement arises naturally from the strategy of remembering a bit using majority functions. We now present a particular definition for an "addressing scheme" along with some supporting definitions.

An *edge-coloring* of a graph G is a map $c : E(G) \to S$ where S is a set of colors such that $c(e_1) \neq c(e_2)$ when e_1 and e_2 are adjacent edges. With an edge-coloring, a path between two vertices can be described by giving the sequence of edge colors encountered along the path. We can directly use an edge-coloring to make an addressing scheme. With an origin vertex specified, the *address* of another vertex is given by a list of colors. The inverse of any path is simply the sequence of colors in reverse order.

Definition 10 (addressing scheme)**.** An *addressing scheme* for a graph is an edge-coloring and a specified vertex from the graph called the *origin*. We call an addressing scheme either *finite* or *infinite* according to the number of colors used by the edge-coloring.

Our results require finite addressing schemes with the following property:

Definition 11 (shortest-path-invariant)**.** We call an addressing scheme *shortest-path-invariant* with respect to a fixed origin if, for every color k and for every vertex a, the number of edges of color k is the same for all shortest paths from the origin to a. In other words, the number of edges of any particular edge color is invariant for any pair of shortest paths from the origin to a vertex.

Notice that it is not true in general that if a shortest-path-invariant addressing scheme exits, then any two valid addresses with the same color counts correspond to the same cell.[13] Figure 3.13 gives an example of an addressing scheme for the regular tessellations $\{4, 5\}$ along with an idea of how its construction is achieved.

[13]Infinite cubic lattices in R^d do have addressing schemes with this property.

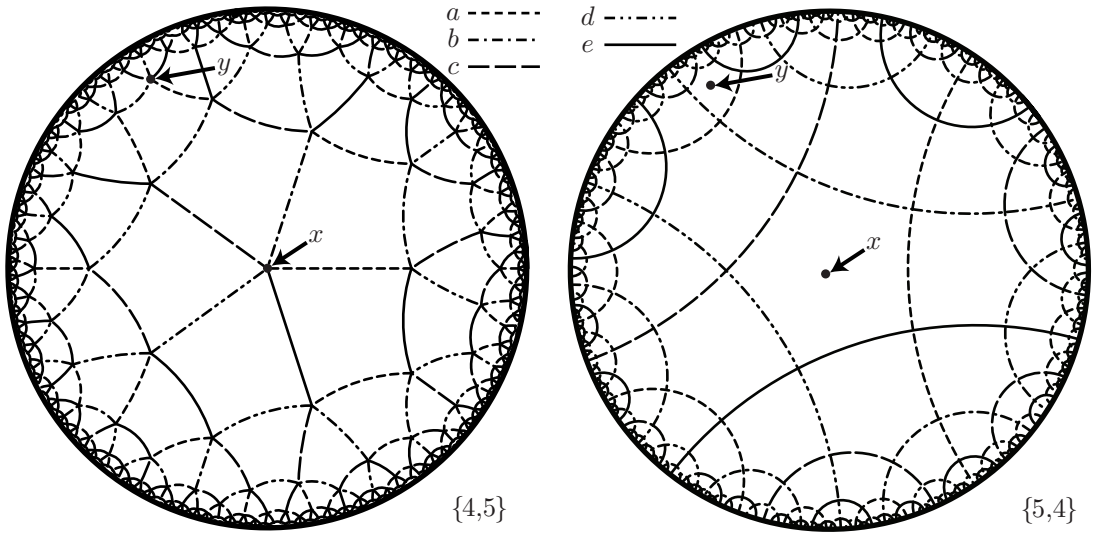

Figure 3.13: The edge coloring of the $\{4,5\}$ graph is obtained from colors of intersecting geodesics in its dual graph ($\{5,4\}$). If x is taken as the origin, then the address of y is either *cab*, *cba*, or *bca*. These are also the three shortest paths to y, the only difference being a permutation of the edges.

Theorem 27. *Every graph in* $(((\langle 3,7\rangle \cup \langle 4,5\rangle) \cap \mathcal{P}_{\text{fnt,par}}) \cup \langle \infty, 3\rangle) \cap \mathcal{Q}_{\text{bdd}}$ *has a finite shortest-path-invariant addressing scheme with respect to any cell chosen as the origin. The addressing scheme requires at most* $5q_{\max} - 6$ *colors where* $q_{\max}$ *is the maximum vertex degree in the graph.*

The proof of Theorem 27 requires a fair number of supporting lemmas and definitions. So as not to interrupt the flow of results in this section, we leave its proof and supporting results to the appendix.

Let A be an automaton and let H be the graph obtained from the medium-dependency-graph $\mathfrak{D}_{\mathbf{M}}(A)$, but with all edges made into undirected edges and all self-loops and multi-edges ignored. Suppose that G is a simple undirected graph. We say that A *is an automaton on a graph* G if H is a subgraph of G. We always assume that $V(H) = V(G)$ and that both H and G are connected graphs.

Given a desired trajectory y, a point $a \in \mathbf{L}$, and a configuration $x_{U(a)} \in X_{U(a)}$ of a's input-points, we say that *"a will be forced into error"* or *"$x_{U(a)}$ forces a into error"* if $\phi_a(x_{U(a)}) \neq y_a$. Equivalently, we could say a was forced into error if $a \in E_{x,y}$ and $a \notin D_x$. Recall that $E_{x,y}$ is the set of points in error (with respect to trajectory y) and D_x is the set of failures (or points where the transition function was not followed).

The following Theorem shows how a finite shortest-path-invariant addressing scheme can be used in combination with Toom's Theorem to show that a trajectory is stable.

Theorem 28. *Let A be an automaton on a graph $G \in \mathfrak{G}$. Suppose that there*

exists a finite shortest-path-invariant addressing scheme for G with a^* as the origin. Further, we suppose that there is no dependence between cousin cells (with respect to a^* as origin). A trajectory y is stable if it is true that for any $a \in \mathbf{L}$ and configuration $x_{U(a)} \in X_{U(a)}$ that forces a into error, there exists distinct points $b, c \in U(a) \cap E_{x,y}$ such that at least one of cells $\underline{b}$ or $\underline{c}$ is a child of $\underline{a}$. Note that $\underline{b}$ and $\underline{c}$ need not be distinct from $\underline{a}$.

Proof. Let R be the number of colors used by the addressing scheme. Therefore, for every point $a \in \mathbf{L}$ the following bounds hold: $|U(a)| \leq R$, and $|\{b : a \in U(a)\}| \leq R$. Let $n = R+1$ and let $r = 1/(2n-1)$. For any point $a \in \mathbf{L}$, and $k \in 1, \ldots, n-1$, let $c_k(a)$ be the number of edges of color k on any shortest path from the origin a^* to $\underline{a}$ in G. Any shortest path will do since the addressing scheme is shortest-path-invariant. For $k \in 1, \ldots, n-1$ define

$$L_k(a) = \frac{-nc_k(a) - t(a)}{2n-1}$$

and define

$$L_n(a) = \frac{n\sum_{k=1}^{(n-1)} c_k(a) + (n-1)t(a)}{2n-1}.$$

Since there are no cousin-edges in the dependency graph, it follows that $c_k(b) - c_k(a) \in \{-1, 0, 1\}$ for any $a, b \in \mathbf{L}$ such that $b \in U_{\mathbf{L}}(a)$. Since $t(b) = t(a) - 1$, we have that $|L_k(b) - L_k(a)| \leq 1$ (the scaling factor $1/(2n-1)$ is designed to keep this distance scaled to at most 1). The L-functions have been defined so that $\sum_{k=1}^{n} L_k(a) = 0$ for all $a \in \mathbf{L}$. We have now shown that the first three conditions of Theorem 26 (Toom's Theorem) hold for automaton A independent of the choice of trajectory y. The final condition in Toom's Theorem depends on y and must be verified for all L-functions. Let $a \in \mathbf{L}$ be a point that will be forced into error. The theorem statement provides at least one point $b \in U_{\mathbf{L}}(a)$ with $x_b \neq y_b$ and such that $\underline{b}$ is a child of $\underline{a}$. This implies that $\sum_{k=1}^{(n-1)} c_k(b) = \sum_{k=1}^{(n-1)} c_k(a) + 1$ and therefore $L_n(b) - L_n(a) \geq r$. For $k \in \{1, \ldots, n-1\}$, function L_k is always satisfied since the theorem statement provides distinct points $b, c \in U_{\mathbf{L}}(a)$ with $x_b \neq y_b$ and $x_c \neq y_c$. Since b and c are distinct, one of them, say d, is such that $c_k(d) \leq c_k(a)$. Since $t(d) = t(a) + 1$, the difference $L_k(d) - L_k(a) \geq r$. Therefore y is stable by Toom's Theorem. □

Note that Theorem 28 works for general automata and general trajectories, although it is best suited for automata with monotone transition rules. Theorem 28 provides an easy way to apply Toom's Theorem by looking for simple properties in the medium-dependency-graph and by considering the transition rule. One difficulty can be in finding a finite shortest-path-invariant addressing scheme, but many graphs are covered already by Theorem 27 as well as automata with media equivalent to $\mathbf{Z}^d$. We provide an example of the application of Theorem 28 in the following poof of Theorem 25.

Proof (of Theorem 25). Let A be a majority-rule automaton on $G \in ((\langle 3,7\rangle \cap \mathcal{P}_{\text{bdd,par}}) \cup (\langle 4,5\rangle \cap \mathcal{P}_{\text{fnt,par}}) \cup \langle \infty, 3\rangle) \cap \mathcal{Q}_{\text{bdd}}$ and let $a^* \in \mathbf{M}$ be an arbitrary origin cell. By Theorem 27, G has finite shortest-path-invariant addressing scheme with respect to a^*. We assume that the desired trajectory y is either $\mathbf{0}$ or $\mathbf{1}$ since "remembering a bit" only requires showing that these two trajectories are stable.

Suppose that $G \in \langle \infty, 3\rangle$. Since G is a tree (with any origin a^* as the root), there are only parent and children relationships between cells. By the lower-bound on the degrees of vertices and by the definition of a majority-rule automata (Definition 6), it true that if any point $a \in \mathbf{L}$ will be forced into error, then there exists at least two distinct points $b, c \in U(a) \cap E_{x,y}$ such that one of $\underline{b}$ or $\underline{c}$ is a child of $\underline{a}$. By Theorem 28, trajectory y is stable.

Suppose that $G \in \langle 4,5\rangle$. In general, there can be cousin-edges between vertices, and therefore there can be dependence between cousin cells in the graph. Let B be a pessimistic version of A (see Definition 8) defined as follows. Let B be identical to A except for the following modification: all cells assume that their cousins (assuming they depend on them) are constantly in error (with respect to y). By Lemma 16, if trajectory y is stable for B, then it is also stable for A. Therefore, we focus just on automaton B. By definition, B satisfies the requirement of Theorem 28 in that cells do not depend on their cousins. By Theorem 10, every vertex $v \in V(G)$ is one of three types: (type I) the vertex has exactly one cousin-neighbor and exactly one parent, (type II) the vertex has no cousin-neighbors and exactly two parents, and (type III) the vertex has no cousin-neighbors and exactly one parent. Assume that point $a \in \mathbf{L}$ will be forced into error. If $\underline{a}$ is a type III cell, then it is trivial to apply Theorem 28. Suppose that $\underline{a}$ is a type I cell. The error-threshold for $\underline{a}$ has been lowered by one (compared to A) by the construction of B. If $d_G(\underline{a})$ is even, then the error-threshold for a is at least three. Even assuming that both the point $\in U(a)$ corresponding to the parent-cell of $\underline{a}$, and the point $(\underline{a}, t(a)+1)$ are in error, there must still be an additional point $\in U(a)$ that corresponds to a child-cell of $\underline{a}$ and is in error. If $d_G(\underline{a})$ is odd, then the error-threshold is one less (than the case where $d_G(\underline{a})$ even), but $\underline{a}$ no longer depends on itself. In both cases, Theorem 28 applies. The final case is when $\underline{a}$ is a type II cell. Since $\underline{a}$ had no cousin-edges to begin with, the error-threshold for $\underline{a}$ is always one higher than when $\underline{a}$ is a type I cell. This increased threshold compensates for a's additional parent, and the analysis goes through almost identically as when $\underline{a}$ is of type I. Therefore, by Theorem 28, trajectory y is stable.

Finally, suppose that $G \in \langle 3,7\rangle$. The $\langle 3,7\rangle$ case requires some special handling since we cannot apply Theorem 28 directly. We still apply Toom's Theorem, and will follow the strategy used in the proof of Theorem 28. By Theorem 38, the periodic fault model is equivalent to the transient fault model[14]. Automaton A, under the periodic fault model with fault-period two, is equivalent to the transient

[14]The proof of Theorem 25 is the only result in the thesis that references a result from a later chapter. The results of Chapter 5 do not rely on any results from elsewhere in the thesis.

model operating on a modified automaton B which "advances" A two timesteps at a time for each timestep in B. Therefore, each cell a in B has $U^2(a)$ (w.r.t. A) as its input-set. Therefore, it is sufficient to apply Toom's Theorem to automaton B under the transient fault model. Except for when noted, assume we are reasoning about automaton B in the following.

The restriction $\mathcal{Q}_{\text{bdd}}$ in the theorem statement implies that there exists a constant $q_{\max}$ that bounds the degree of all vertices in G. Therefore, each cell in B depends upon, and is depended upon by, at most $R = (q_{\max} + 1)^2$ vertices. Thus, condition (1.) of Toom's Theorem is satisfied.

For any point $a \in \mathbf{M}$, and $k \in 1, \ldots, n-1$, let $c_k(a)$ be the number of edges of color k on any shortest path from the origin a^* to a in G. Any shortest path will do since the addressing scheme is shortest-path-invariant. Unlike the case $G \in \langle 4, 5\rangle$, for the case $G \in \langle 3, 7\rangle$, we cannot assume that cells ignore their cousins. Therefore, we must use the restriction on face[15] degrees in G provided by restriction $\mathcal{P}_{\text{bdd,par}}$ on $\langle 3, 7\rangle$. Let $p_{\max}$ be the maximum face degree. Restriction $\mathcal{P}_{\text{par}}$ implies that all faces are of either odd- or even-degree parity. Observe that $\langle 3, 7\rangle \cap \mathcal{P}_{\text{bdd,evn}} \cap \mathcal{Q}_{\text{bdd}} \subset \langle 4, 7\rangle \cap \mathcal{Q}_{\text{bdd}}$, and recall that majority-rule automata on graphs in $\langle 4, 7\rangle \cap \mathcal{Q}_{\text{bdd}}$ were shown to be able to remember a bit under the (stronger) combined fault model by Theorem 20. Therefore, we assume that the parity of all faces is odd. It can be shown that any two vertices $a, b \in V(G)$ which are at most shortest-path distance γ apart have a common ancestor at a distance that is bounded as a function of γ and $p_{\max}$. The distance between any cells a and b (in automaton B) such that $b \in U_{\mathbf{M}}(a)$ is at most $\gamma = 2$. Therefore, let the bound on the distance to a nearest common ancestor be σ (the actual value of σ is not important). Then $|c_k(a) - c_k(b)| \leq \sigma$ for all $a, b \in \mathbf{M}$ where $b \in U_{\mathbf{M}}(a)$. For all $a, b \in \mathbf{L}$, if $b \in U_{\mathbf{L}}(a)$ then $t(b) = t(a) + 1$. Let $r = 1/((\sigma + 1)n - 1)$. For $k \in 1, \ldots, n-1$, and $a \in \mathbf{L}$, define

$$L_k(a) = \frac{-nc_k(\underline{a}) - t(a)}{(\sigma + 1)n - 1}$$

and define

$$L_n(a) = \frac{n\sum_{k=1}^{(n-1)} c_k(\underline{a}) + (n-1)t(a)}{(\sigma + 1)n - 1}.$$

For all $1 \leq k \leq n$, if $b \in U_{\mathbf{L}}(a)$, then $|L_k(b) - L_k(a)| \leq 1$ (note that $1/((\sigma + 1)n - 1)$ was chosen to keep this difference scaled to at most 1). Thus, condition (2.) of Toom's Theorem is satisfied. Since the L-functions are defined such that $\sum_{k=1}^{n} L_k(a) = 0$ for all $a \in \mathbf{L}$, condition (3.) of Toom's Theorem is satisfied.

We now show that condition (4.) of Toom's Theorem is satisfied. Let $a \in \mathbf{L}$ be a point that is about to be forced into error by the points $S = U_{\mathbf{L}}(a) \cap E_{x,y}$.

[15]Recall that we assume there is a particular plane-embedding for G which makes the notion of "face" well-defined. We can view the embedding of G in the plane as defining the faces of an infinite polyhedron.

The functions $L_1, \ldots, L_{n-1}$ have been designed so that if we can show that for every $k \in \{1, \ldots, n-1\}$, there exists $b \in S$ such that $c_k(\underline{b}) \leq c_k(\underline{b})$, then condition (4.) is satisfied for these L-functions. This is easily verified, but we first need to get a handle on the cases we need to consider. Lemmas 8 and 9, and Theorem 12 show that all vertices have either one or two parents, at most two cousins, and at least two children. For a moment, it will be better to think about automaton A under the periodic fault model. Observe that every cell a depends (w.r.t. A) on at least two non-cousin vertices c and d since the threshold for every cell (w.r.t. A) is at least four. Therefore, with respect to any color k, only one of c or d can have a higher color count in that color. This property also applies to c and d. Therefore, if a will be forced into error (w.r.t. A under the periodic fault model with fault-period two), then it depends on at least one cell in $U^2(a)$ that is not larger in the kth color count. Returning to automaton B, there must exist $b \in S$ such that $c_k(b) \leq c_k(a)$. Thus, condition (4.) is satisfied for $L_1, \ldots, L_{n-1}$. The final L-function, L_n has been designed so that if we can show that there exists $b \in S$ such that $\sum_{k=1}^{n-1} c_k(\underline{b}) > \sum_{k=1}^{n-1} c_k(\underline{a})$ (i.e. $\underline{b}$ is further from a^* than is $\underline{a}$), then condition (4.) is satisfied for L_n. In this case, cousin neighbors (when they exist) work to our advantage. Again, it will, momentarily, be better to think about automaton A under the periodic fault model with fault-period two. Suppose, as is the worst case, that a has two parents and two cousin-inputs and an error-threshold of four. By properties of $\langle 3, 7 \rangle$ summarized in Lemmas 8 and 9, and Theorem 12, it can be shown that the cousin-inputs of a both have at most one parent. It can easily be verified with a few cases that $b \in S$ exists such that $\sum_{k=1}^{n-1} c_k(\underline{b}) > \sum_{k=1}^{n-1} c_k(\underline{a})$. Thus condition (4.) is satisfied, and the entire proof is finished. $\square$

Chapter 4

Quantitative Considerations

This chapter is devoted to more quantitative, rather than qualitative results. The following section presents two results that establish asymptotically near-matching upper and lower bounds on tolerable fault-rates in terms of bounds on the number of neighbor cells in an automaton. The first result is a negative result that gives a lower bound on the distance that the fault rate must be from 1/2 and still be fault tolerant (either transient or combined faults). The second result is a positive result that gives an upper bound by showing the existence of automata (with various number-of-neighbor-cell bounds) that can remember a bit with combined fault-rates that approach 1/2. The final section presents an infinite family of finite-sized, number-of-cell-neighbors-bounded automata that can remember a bit for the theoretically longest possible time (asymptotically speaking). These automata are based on expander graphs and provide a good finite analogue to our results on hyperbolic tessellations.

4.1 Fault Rates

The combined fault model with an adversary is computationally weaker than the transient or manufacturing fault models with adversaries. The transient fault model with adversary is computationally weaker than the probabilistic transient fault model. However, it is not true in general that the manufacturing fault model with adversary is computationally less powerful than the probabilistic transient fault model (consider these two fault models for an automaton consisting of a single cell for example). We are most interested in the combined and transient fault models and these are both covered by the probabilistic transient fault model.[1] As such, for the negative results in this section, we assume the use of the probabilistic transient fault model.

[1]When the medium-dependency-graph for an automaton has the property that any pair of paths between two cells have the same length, then the negative results found under the probabilistic transient fault model also hold for the manufacturing fault model on this automaton.

In this section, we seek conditions under which an automaton cannot remember a bit. Let $\mathcal{A}_k$ be the set of all binary automata in which every cellular transition function depends on at most k inputs. For $A \in \mathcal{A}_k$, let $\gamma(A)$ be the distance of the threshold from $1/2$, such that if the fault-rate $\varepsilon > 1/2 - \gamma(A)$, then the automaton cannot remember a bit. Let $\gamma(\mathcal{A}_k) = \inf_{A \in \mathcal{A}_k} \gamma(A)$. We will show that $\gamma(\mathcal{A}_k) = \Omega(1/\sqrt{k})$ and $\gamma(\mathcal{A}_k) = O(1/\sqrt{k/\log k})$.

We begin with the negative result. Our proof that $\gamma(\mathcal{A}_k) = \Omega(1/\sqrt{k})$ follows from a negative result for noisy circuits of Evans and Schulman in [9] (which is based on Evans' Ph.D. thesis [8]). In [9], a noisy circuit is a circuit composed of gates that fail with probability (exactly) ε (where $\varepsilon \in (0, 1/2)$).[2] They prove the following

Lemma 29 (Lemma 2, Section 1 of Evans and Shulman in [9])**.** *Let G be a circuit composed of ε-noisy gates. Suppose each input to G is X (a binary random variable) or a constant. Let W be the vector of random values carried by a set of wires in G. Then*

$$I(X; W) \leq \sum_{P \text{ from } X \text{ to } W} (1 - 2\varepsilon)^{2|P|}$$

where the sum is over paths P in G from input X to wires in W, and $|P|$ is the number of gates on the path P.

The quantity which the sum bounds is the mutual information between random variables X and W. For our purposes, W is just a single binary random variable, say Y, representing a single output wire. The mutual information between X and Y is defined as

$$I(X; Y) = H(X) - H(X|Y)$$

where $H(X)$ is the entropy or self-information of X (as first defined by Shannon) and $H(X|Y)$ is the conditional entropy of X given Y.[3] It is not difficult to see that $I(X; Y) = I(Y; X)$. Intuitively, $I(X; Y)$ measures the difference between our uncertainty about X and our uncertainty about X given that we know Y. If X and Y are independent, then Y tells us nothing about X and $H(X|Y) = H(X)$ and $I(X; Y) = 0$. At the other extreme, if X determines Y, then $H(X|Y) = 0$ and $I(X; Y) = H(X)$. In general, if Y is not independent of X and if X does not determine Y, then $0 < I(X; Y) < H(X)$.

Theorem 30. *Let A be a binary automaton where the transition function of each cell depends on at most k inputs. Then A cannot remember a bit when the probabilistic transient model fault-rate ε exceeds $1/2 - 1/(2\sqrt{k})$.*

[2] In [9] the term "err" is used rather than "fail," but we will stick with our definitions for failure and error to avoid confusion. They also write $(1 - \xi)/2$ for the failure rate, but we will stick with our convention of using ε to avoid confusion.

[3] Recall that $H(X) = -\sum_x \Pr[X = x] \log_2 \Pr[X = x]$ and $H(X|Y) = -\sum_{x,y} \Pr[X = x, Y = y] \log_2 \Pr[X = x|Y = y]$.

Proof. For any binary cellular automaton, we can view the state of a cell a at time n as a binary random variable ξ_n. We are interested in the ability of the automaton to remember the **0** and **1** trajectories. Therefore, let ζ be a binary random variable that indicates the bit to be remembered. To initialize the automaton, all boundary points are set to ζ. We can view ξ_n as the output of a circuit of depth n (give or take a constant, but this is unimportant here) which has ζ as input. If the automaton is able to remember a bit, then $I(\zeta;\xi_n)$ remains bounded away from 0. In other words if $\lim_{n\to\infty} I(\zeta;\xi_n) = 0$, then the automaton cannot remember a bit[4]. When we view ξ_n as the output of a circuit based on input ζ, we find that the probabilistic transient fault model is equivalent to the ε-noisy circuit model. The circuit G connecting input ζ to output ξ_n has the property that any path from ζ to ξ_n contains exactly n gates (these n "gates" correspond to the transition functions of n points). Furthermore, each gate has at most k inputs by the assumption that $A \in \mathcal{A}_k$. There are at most k^n different paths from ζ to ξ_n, and therefore, by Lemma 29

$$I(\zeta;\xi_n) \leq k^n(1-2\varepsilon)^{2n}.$$

This implies that automata cannot remember a bit when

$$\varepsilon > \frac{1}{2} - \frac{1}{2\sqrt{k}}.$$

□

The argument in [9] and [8], follows the development of a similar type of negative result by Pippenger in [25]. Pippenger uses information theoretic techniques to obtain a lower bound on formula depth (a formula being a circuit without fanout one at gates). In [10], Feder generalizes Pippenger's result to circuits using a probabilistic technique. Evans and Schulman's argument provides sharper lower bounds than Pippenger and Feder. Using Feder's results, we would only be able to only conclude that automata in $\mathcal{A}_k$ cannot remember a bit when their fault-rate is $> 1/2 - 1/(2k)$.

To obtain an upper-bound on $\gamma(\mathcal{A}_k)$, we describe a family of automata $\mathcal{B} = \bigcup_k \mathcal{B}_k$ where for every k (sufficiently large), $\mathcal{B}_k \subset \mathcal{A}_k$ and $B \in \mathcal{B}_k$ can remember a bit when the fault-rate is $\leq 1/2 - 1/\sqrt{k/\ln k}$.

The following result is the first result (of which we are aware) that gives a construction for an automaton that can remember a bit for any given fault-rate that is arbitrarily close to 1/2. Our automata work equally well under the transient or combined fault models. The proof of Toom's Theorem [30, Sec. 1, Theorem 1], although showing stability for some of our automata (his theorem does not cover automata with unbounded in-degrees), actually requires increasingly smaller fault-rates as the medium dependency-graph vertex degrees increase.

[4]Fano's inequality makes this precise. Gallager's text [15] provides good background on information theory.

This is likely because Toom's Theorem was formulated with finite-dimensional Euclidean spaces in mind, and the theorem is only concerned with the stability as the fault-rate approaches zero.

Theorem 31. *Suppose that we are given $0 < \delta < 1/2$. Let A be a binary automaton for which both $\mathbf{0}$ and $\mathbf{1}$ are trajectories and let either $y = \mathbf{0}$ or $y = \mathbf{1}$ be the desired stable trajectory. Suppose that the medium dependency-graph of A is a forest. For each $a \in \mathbf{M}$, let $h(a)$ represent the minimal number of inputs in error necessary to force ϕ_a into error ($h(a)$ must be a lower bound for both $y = \mathbf{0}$ and $y = \mathbf{1}$). Let $d = \min_{a \in \mathbf{M}} |U(a)|$ and suppose that there exists a constant m such that $|U(a)| \leq 2h(a) + m$ for all $a \in \mathbf{M}$. If*

$$d \geq m + 2(\ln 2)(m + 1 - \log_2 \delta)/\delta^2,$$

then A can tolerate a fault-rate of $1/2 - \delta$ under the combined fault model.

Proof. Let the fault rate ε be at most $1/2 - \delta$, and let $P_n = \sup_{a \in \mathbf{L}_n} \Pr[a \in E_{x,y}]$ for $n \geq 0$. We shall prove by induction on n that P_n is at most $1/2 - \delta/2$. For the base case, note that $P_0 = \varepsilon \leq 1/2 - \delta < 1/2 - \delta/2$. For the inductive step, note that since $P_n < 1/2$, we have $P_n < 1 - P_n$, $2(1 - P_n) > 1$, and $4P_n(1 - P_n) < 1$.

Then, for any $a \in L_{n+1}$ with $n \geq 0$,

$$\Pr[a \in E_{x,y}] \leq \varepsilon + (1 - \varepsilon) \sum_{\substack{A \subseteq U(a), \\ |A| \geq h(a)}} \Pr[(b \in E_{x,y},\, \forall b \in A) \wedge (b \notin E_{x,y},\, \forall b \in U(a) - A)] = (*)$$

The event represented by $[(b \in E_{x,y}, \forall b \in A) \wedge (b \notin E_{x,y}, \forall b \in U(a) \setminus A)]$ is stochastically dominated by an event where each of a's neighbors is independently in error with probability exactly P_n (rather than at most P_n). We therefore get the following expressions as upper-bounds on $(*)$:

$$\begin{aligned}
(*) &\leq \varepsilon + \sum_{\substack{A \subseteq U(a), \\ |A| \geq h(a)}} {P_n}^{|A|}(1 - P_n)^{|U(a)| - |A|} \\
&= \varepsilon + \sum_{k=h(a)}^{|U(a)|} \binom{|U(a)|}{k} {P_n}^{k}(1 - P_n)^{|U(a)| - k} \\
&\leq \varepsilon + {P_n}^{h(a)}(1 - P_n)^{|U(a)| - h(a)} \sum_{k=h(a)}^{|U(a)|} \binom{|U(a)|}{k} \\
&\leq \varepsilon + {P_n}^{h(a)}(1 - P_n)^{|U(a)| - h(a)} 2^{|U(a)|} \\
&= \varepsilon + {P_n}^{h(a)}(1 - P_n)^{-h(a)}(2(1 - P_n))^{|U(a)|} \\
&\leq \varepsilon + {P_n}^{h(a)}(1 - P_n)^{-h(a)}(2(1 - P_n))^{2h(a)+m} \\
&= \varepsilon + 2^m(1 - P_n)^m(4P_n(1 - P_n))^{h(a)} \\
&\leq \varepsilon + 2^m(4P_n(1 - P_n))^{(d-m)/2}.
\end{aligned}$$

To clarify the algebra, let $c = (d - m)/2$. Since the right side of the above inequality does not depend on a but only on n, we have

$$P_{n+1} = \sup_{a \in L_n} \Pr[a \in E_{x,y}] \leq \varepsilon + 2^m (4P_n(1 - P_n))^c.$$

To finish the proof by induction, we need to find a lower bound on c such that

$$P_{n+1} \leq \varepsilon + 2^m (4P_n(1 - P_n))^c \leq 1/2 - \delta/2.$$

Given that $P_n \leq 1/2 - \delta/2$ and $\varepsilon \leq 1/2 - \delta$, it is sufficient to find c such that

$$(1 - \delta^2)^c \leq \delta/2^m.$$

By taking logarithms and using the bound $\log_2(1 - \delta^2) \leq -\delta^2/\ln 2$, we find that the induction holds when

$$c \geq (\ln 2)(m + 1 - \log_2 \delta)/\delta^2.$$

Substituting $c = (d - m)/2$ gives

$$d \geq m + 2(\ln 2)(m + 1 - \log_2 \delta)/\delta^2.$$

Therefore, $\sup_{a \in \mathbf{L}} \Pr[a \in E_{x,y}] \leq 1/2 - \delta/2$ when $\varepsilon \leq 1/2 - \delta$ and, by definition, y is a stable trajectory for automaton A with fault rate $\varepsilon \leq 1/2 - \delta$. □

In Theorem 31, we require that $|U(a)| \leq s\, h(a) + m$ where $s = 2$. If we know $s < 2$, then we can derive a better bound on d, but such bounds on $s < 2$ do not correspond to any transition functions in which we are interested, and so we do not bother to carry s through the calculations. If $s > 2$, then the technique used above cannot work since we would get $(2^s P_n(1 - P_n))^c$, but as $P_n \to 1/2$, this requires that s be arbitrarily close to 2 so that $2^s P_n(1 - P_n) < 1$.

Corollary 32. *Let A be a binary automaton for which both $\mathbf{0}$ and $\mathbf{1}$ are trajectories and let either $y = \mathbf{0}$ or $y = \mathbf{1}$ be the desired stable trajectory. Suppose that the medium dependency-graph of A is a forest. For each $a \in \mathbf{M}$, let $h(a)$ represent the minimal number of inputs in error necessary to force ϕ_a into error ($h(a)$ must be a lower bound for both $y = \mathbf{0}$ and $y = \mathbf{1}$). Suppose there exists constants m and d such that $d \leq |U(a)|$ and $|U(a)| \leq 2h(a) + m$ for all $a \in \mathbf{M}$. Let*

$$\delta = \sqrt{\frac{2(\ln 2)(m + 1) + \ln(d - m)}{d - m}}.$$

If $0 < \delta < 1/2$, then y is stable under the combined fault model with fault rate $\varepsilon = 1/2 - \delta$.

Proof. To prove this corollary, we need to invert the inequality of Theorem 31. Doing this we get

$$\delta \geq \sqrt{\frac{W_p(2^{2(m+1)}(d-m))}{d-m}}.$$

where W_0 is defined (as in [4]) as follows. Let W be the Lambert W function (or product log) which is the multivalued inverse of function $w \mapsto we^w$. When the argument to W is a real $\geq -1/e$, and if we require $w > -1$, then W is a single-valued function ranging over the reals called the *principal branch* of the W function and is denoted W_0. Noting that $W_0(x) \leq \ln x$ when $x \geq e$, we get the expression for δ given in the theorem statement. □

Corollary 33. *For every $q \geq 81$, if $\delta = \sqrt{18 \ln 2 + \ln(q-8)}/\sqrt{q-8}$ then all majority-rule automata on graphs in $\langle 3, q\rangle$ can tolerate combined faults with fault rate $0 < \varepsilon = 1/2 - \delta$.*

Proof. By the proof of Theorem 19, for $q \geq 14$, we can construct a pessimistic version B of any automaton A on a graph in $\langle 3, q\rangle$ with the properties: (1) B's medium dependency graph is a tree, and (2) $d = q-6 \leq |U(a)| \leq 2h(a)+8$ for all cells a in automaton B. By Lemma 16, A is tolerant of combined faults at a particular fault rate whenever B is. By Corollary 32, if $\delta = \sqrt{18 \ln 2 + \ln(d-8)}/\sqrt{d-8}$ then B can tolerate combined faults with fault rate $\varepsilon = 1/2 - \delta$. If $q \geq 80$, the $\delta > 0$, and automaton A can tolerate combined faults with fault-rate $0 < \varepsilon = 1/2 - \delta$. □

At the beginning of this section, we defined the set of automata $\mathcal{A}_k$ and function γ. It follows immediately from Theorem 30 that $\gamma(\mathcal{A}_k) = \Omega(1/\sqrt{k})$, and from Corollary 33 that $\gamma(\mathcal{A}_k) = O(1/\sqrt{k/\log k})$.

4.2 Automata on Expanders

An argument for studying automata with infinite cells is that it gives an indication of the type of behavior to expect in finite automata with similar characteristics. We have many many results for automata whose dependency graphs are trees or are tree-like. For many such automata, we find that they can remember a bit for an infinite amount of time with combined faults. Additionally, we have shown that a large, and natural, class of automata with medium dependency graphs on two-dimensional Euclidean lattices are unable to remember a bit. Restricted to a finite medium, we wish to find automata that can remember a bit under combined faults with an expected-value time exponential in the number of cells.

In [24, p. 33], Pippenger describes a gadget called a "compressor." A (m, k, α, β)-*compressor* is a bipartite multigraph with m inputs, m outputs, and k edges incident with each input and output. Additionally, it has the following property:

for every set A containing at most αm inputs, at most βm outputs are connected to at least $k/2$ inputs in A. Using bipartite multigraph expanders of Jimbo and Maruoka[17], Pippenger proves the following lemma (Lemma 3.2 in [24]) concerning the existence of an infinite family of compressors.

Lemma 34. *For every $m = p^2$ (p integral), there is a $(m, 8^{17}, 1/64, 1/512)$-compressor. Furthermore, its incidence matrix can be computed in space $O(\log m)$.*

The compressor property follows from the spectral properties of the graph. Lemma 34 is used in [24] in a theorem about reliably computing Boolean functions with noisy circuits. Following a similar approach, we use the lemma to suppress errors in automata with finite number of cells.

Theorem 35. *Let $c = (e^6/7^7)^{1/512} \approx 0.985$, and let the fault rate of automata under the combined fault model be at most $1/512$. For every $m = p^2$ (p integral), there exists an automaton with m cells and with constant degree inputs and outputs that can remember the "all zeros" and "all ones" initial state for at least $1/c^{m/2}$ timesteps (except for with probability $c^{m/2}$).*

Proof. By Lemma 34, there is a $(m, 8^{17}, 1/64, 1/512)$-compressor with an explicit construction. Let $\mathbf{M} = \{a_1, \ldots, a_m\}$ be the cells of the automaton. Each cell a_i has a transition function ϕ_{a_i} which is an 8^{17}-input majority gate[5]. The medium $\mathbf{M}$ acts as both the input and output sets of vertices in the bipartite graph defined by the compressor. The input cells to each ϕ_{a_i} is defined according to the input-set-to-output-set edges defined by the compressor.

Initially, the cells are set to either the "all zeros" or "all ones" initial state. We say that automaton has "not yet failed" so long as not more than $m/64$ cells have been in error at any given timestep from initialization to the present timestep. When the first time that $> m/64$ cells are in error occurs, we say that the automaton has "failed." We wish to estimate the first time to failure. Suppose that at most $m/64$ cells at timestep n are in error (after the faults for that timestep have taken effect). Then at the start timestep $n + 1$, there are at most $m/512$ cells in error. This is true by the property of the compressor that there are at most $m/512$ outputs that receive at least half of their inputs from the $m/64$ inputs in error. At each timestep, faults occur at each cell independently[6] with probability at most $1/512$. So long as no more than $(7m)/512$ faults occur at a given timestep, then at most $m/64$ cells are in error at that timestep. Using the Chernoff bound [22, Ch. 4], at most $7m/512$ faults occur at each timestep

[5]The output from the majority function when the number of zero inputs equals the number of one inputs can be chosen arbitrarily.

[6]Since there are manufacturing faults present, the probability of a fault at a cell from one timestep to the next is not independent. However, amongst cells at the same timestep, the probability of there being faults (transient or manufacturing) at every cell in a subset A of cells is $\leq \varepsilon^{|A|}$ where ε is the combined fault rate.

except with probability at most $(e^6/7^7)^{m/512} = c^m$. By the union bound, the first-time-to-failure is after at least $1/c^{m/2}$ timesteps, except for with probability $c^{m/2}$. Therefore, with high probability, the automaton remembers its initial state for a time exponential in its size. □

Since [24], the state of the art in expander graphs has improved considerably. Different expanders could be used to construct more realistic automata with better bounds, but we are mainly interested in the existence of a class of automata like the family described in Theorem 35. See the recent paper [16] of Hoory, Linial, and Wigderson for a good survey on expander graphs.

Chapter 5

Other Fault Models

There are a number of interesting variant fault models to consider. Notice that manufacturing faults are a type of permanent fault; when a manufacturing fault occurs at a cell, that cell is forever under the control of the adversary. Therefore, it is important that the probability of a permanent fault at any given cell is bounded for all time. If permanent faults are allowed to accumulate, then the probability of any given cell being under the control of the adversary is arbitrarily close to one after a sufficient number of timesteps. In this case, there clearly needs to be some mechanism to fix permanently broken cells. Such a permanent-fault repair mechanism would seem to require a significantly different type of fault model and we consider it beyond the scope of this thesis. The initial-round manufacturing fault model is the strongest type of permanent fault mechanism we consider.

Between the extremes of permanent and transient faults, we can describe a class of faults called *persistent faults*. Persistent faults are like transient faults in that they can occur at a cell at any timestep, but they are also like permanent faults because their duration is longer than one timestep. The critical attribute of persistent faults is that their expected duration (the interval of time during which the adversary has control over the cell) is finite.

One persistent fault model we consider we call the *delayed-repair fault model*. In this model, transient faults occur independently at each point in the lattice with a bounded probability, but their effect is for a fixed number of timesteps which we call the *fault duration* parameter.

It is obvious that the delayed-repair fault model "dominate" the transient fault model in the following sense: if a trajectory is stable under the delayed-repair fault model, then the trajectory is stable under the transient fault model. We make this notion formal by saying that one fault model *dominates*[1] the other fault model if stability under the one implies stability under the other. Domination is a transitive relation, and therefore, if two fault models dominate each other,

[1]Our definition of "domination" differs from the usual definition for "stochastic domination." However, we do use implicitly stochastic domination in the proofs of domination between fault models.

then we call them *equivalent* fault models. Note that stability is defined as the probability of error going to zero as the fault-rate goes to zero. For a given fault-rate, cells in an automaton may be more likely to be in error under fault model A than under fault model B, but it is entirely possible that fault model B dominates fault model A in our technical sense.

Theorem 36. *The transient fault model is equivalent to the delayed-repair fault model.*

Proof. Let m be the fault duration for the delayed-repair model. Let β_1 be the fault rate for the delayed-repair fault model, and let β_2 be the fault rate for the transient fault model. Suppose that $\beta_1 < {\beta_2}^m$. For any $(a, n) \in \mathbf{L}$, it is more probable that $\{(a, n), (a, n+1), \ldots, (a, n+m-1)\} \subset F_T$ under the transient fault model than $(a, n) \in F_T$ under the delayed-repair fault model. By this reasoning, the transient fault model dominates the delayed-repair fault model. Domination in the other direction is trivial and therefore the fault models are equivalent. □

Whereas it is obvious that persistent faults only have the potential to help the adversary, we can describe a fault model that appears to be less advantageous for the adversary. In the *periodic fault model*, faults only occur at fixed multiples of timesteps rather than on every timestep. We call the timesteps where faults occur *fault-timesteps*, and we call the timesteps when they do not occur *faultless-timesteps*. The period for fault timesteps is called the *fault-period*. Clearly, the transient fault model is dominated by the periodic fault model. Showing that the converse holds is somewhat more difficult. To do this, we need to qualify the result with a slight restriction on the class of automata considered: we require that the in-degree and out-degree of each cell is bounded. We first present a supporting lemma.

Lemma 37. *Let A be an automaton where every cell has a cell-input set of bounded size. Let m be a positive integer and let y be a trajectory. If*

$$\lim_{\beta \to 0} \sup_{\substack{a \in \mathbf{L} \text{ s.t.} \\ t(a) \equiv 0 \,(\mathrm{mod}\, m)}} (\mu_0 \times \nu_\beta)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}) = 0, \tag{5.1}$$

then

$$\lim_{\beta \to 0} \sup_{a \in \mathbf{L}} (\mu_0 \times \nu_\beta)(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}) = 0.$$

Proof. Let q be the bound on cell-input sets (the number of cells on which the transition function depends). For a particular fault-rate β, suppose that $\Pr[a \in E_{x,y}] \leq P_n$ for all $a \in \mathbf{L}_n$. Then $\Pr[a \in E_{x,y}] \leq qP_n + \beta$ for all $a \in \mathbf{L}_{n+1}$. After m timesteps, we get the bound $\Pr[a \in E_{x,y}] \leq q^m P_n + \beta(q^m - 1)/(q - 1)$ for all

$a \in \mathbf{L}_{n+m}$. Therefore, as the bound on errors over all points at multiple-of-m timesteps and the fault-rate β both tend to zero, the bound on errors over all points at all timesteps tend to zero. □

Theorem 38. *The periodic fault model is equivalent to the transient fault model when restricted to automata where every cell has bounded cell-input and cell-output sets.*

Proof. Let m be the fault-period for the periodic fault model and let $q-1$ be the in- and out-degree bound on cells in the automaton. Let β_1 be the fault rate for the transient fault model and β_2 be the fault rate for the periodic fault model. Suppose that $\beta_1 < {\beta_2}^{q^m}$. Consider any finite subset of points A on $\mathbf{L}_{n+1} \cup \mathbf{L}_{n+2} \cup \cdots \cup \mathbf{L}_{n+m}$ where n is a timestep that is a multiple of m. Under the transient fault model, the probability that $A \subset F_T$ is ${\beta_1}^{|A|} < {\beta_2}^{q^m |A|}$. Since each cell is depended upon by at most q cells from timestep to timestep, at most $q^m |A|$ points in $\mathbf{L}_{n+m}$ are influenced by the points in A. The probability that any set of $q^m |A|$ points is in the fault set under the periodic fault model is ${\beta_2}^{q^m |A|}$. Therefore, $\Pr[a \in E_{x,y}]$ for timestep $n \equiv 0 \pmod m$ under the transient fault model (with fault rate β_1 less than ${\beta_2}^{q^m}$) is bounded by $\Pr[a \in E_{x,y}]$ for timestep $n \equiv 0 \pmod m$ under the periodic fault model (with fault rate β_2). Since y is a stable trajectory under the periodic fault model, for the transient fault model we can conclude that:

$$\lim_{\beta_1 \to 0} \sup_{\substack{a \in \mathbf{L} \text{ s.t.} \\ t(a) \equiv 0 \,(\mathrm{mod}\, m)}} (\mu_0 \times \nu_{\beta_1})(\{(F_M, F_T) : \exists x \in C_y \text{ s.t. } a \in E_{x,y} \wedge D_x \subseteq F_C\}) = 0.$$

Since every cell has a bounded sized cell-input set, we conclude by Lemma 37 that y is a stable trajectory under the transient fault model. Therefore, the periodic fault model is equivalent to the transient fault model. □

Theorem 38 is particularly useful because it is sometimes easier to demonstrate stability using a seemingly weaker fault model. For example, in the proof of Theorem 25, we use this theorem[2] to enable the use of Toom's Theorem to show transient fault tolerance for majority-rule automata on graphs in $\langle 3, 7 \rangle$.

[2]Note that the results in this chapter did not use any results from elsewhere in the thesis. Theorem 25 is the only result that uses a result from a later chapter.

Chapter 6

Conclusion

We mention a few open problems and some conjectures.

As a general open problem, we note that many of our results are restricted to monotone binary cellular automata, and to the problem of stability for the **0** and **1** trajectories (remembering a bit). It is likely that some of these restrictions can be dropped for certain results. However, we often introduced such restrictions so that we could rely on the adversary having an optimal (greedy) strategy. Therefore, dropping the monotone and binary transition-rule restriction will likely require significantly different types of arguments.

In section 2.1, we proved a negative result for remembering a bit with combined faults by binary monotone automata on the lattice $\mathbf{Z}^2$. We conjecture that this results holds if the transition rule is specified for any finite combination of two-dimensional (Euclidean) lattices (for example, the honeycomb lattice).

In Chapter 3, we gave the complete classification of transient and combined fault tolerant majority-rule automata when the cellular-medium is a regular hyperbolic tessellation $\{p, q\}$. All of the results related to this classification were given for the (very general) classes of hyperbolic tessellations $\langle p, q\rangle$. For a few values of p and q (always "edge cases"), we found it necessary to restrict the class $\langle p, q\rangle$ in various ways. We believe that almost all such restrictions are merely technical requirements in our proofs. Chapter 3 introduced the idea of a finite shortest-path-invariant addressing scheme as a way to provide addresses to cells on a (general) tessellation. These addressing schemes were obtained by showing the existence of graph edge-colorings that satisfy special properties. Besides the two conjectures concerning addressing schemes (mentioned at the end of the appendix), it would be interesting to explore the usefulness of these addressing schemes for other problems. In [21], Margenstern describes a different sort of addressing scheme for the $\{5, 4\}$ tessellation which he calls the "pentagrid." His addressing scheme is tailored to the more general task of performing general computation on the pentagrid (in the absence of faults). Our addressing schemes are tailored for the specific task of capturing shortest paths between cells, but our schemes are very general, even applying to the general classes of hyperbolic

tessellations $\langle p, q \rangle$[1].

In Chapter 4, we gave lower and upper bounds for the threshold distance (in terms of a bound on cell-neighborhood size) of the fault rate[2] from 1/2, for which all binary automata cannot remember a bit. The upper and lower bounds agree to within a logarithmic factor, and it is unclear to us what the correct order of magnitude is.

[1]We conjecture that the restriction of bounded-degree faces on $\langle p, q \rangle$ (as required by Theorem 27 in the appendix) can be removed.

[2]The transient fault rate, but the combined fault rate is implied since this is a negative-type result.

Appendix A

Addressing Schemes

The purpose of the Appendix is to prove Theorem 27 (proof on page 81) which is a theorem concerning the existence of finite graph edge-colorings (see page 44) with particular properties on certain subsets of graphs in $\mathfrak{G}$ (Definition 5, page 19). An edge-coloring for a graph can be used to define an addressing-scheme (Definition 10, page 44). We are interested in finding finite shortest-path-invariant (Definition 11, page 44) addressing-schemes in particular. All of the results in the Appendix are graph theoretic, with little mention of automata, and no mention of fault models. The only references out of this section (excluding the aforementioned definitions) are to the graph theoretic results in Section 3.1, with which we assume familiarly.

The remainder of the results in this section are for graphs embedded in the plane. Following Coxeter and Moser in [6], we view simple, infinite, connected, undirected, plane-embedded graphs as a type of generalized polyhedra. Our definition of the set of graphs $\mathfrak{G}$ provides the basis of our definition.

Definition 12 (polyhedron)**.** A graph $G \in \mathfrak{G}$ is called a *polyhedron* if its planar dual[1] is also in $\mathfrak{G}$.

A polyhedron will often be denoted by $\mathcal{H}$ and we may write $V(\mathcal{H})$ and $E(\mathcal{H})$ to denote the set of vertices and edges respectively in the polyhedron.

Definition 13 (even-face-degree polyhedron)**.** A polyhedron is called an *even-face-degree polyhedron* if all faces have either even or infinite degree.

Definition 14 (edgeset)**.** Let $\mathcal{H}$ be an even-face-degree polyhedron. Since faces are finite, and every face has an even number of edges, the notion of "opposing-edges" is well defined. The relation "opposing-edge" naturally defines an equivalence relation on the set of edges. An *edgeset* is a set of all edges that are equivalent under the opposing-edges relation.

[1] see Section 4.5 in [7] for a construction of the planar dual

Let A and B be edgesets of $\mathcal{H}$. We say that A *intersects* B at a face if there exists non-opposing-edges $a \in A$ and $b \in B$ that bound a common face. If A intersects itself at a face, then A is said to be *self-intersecting* at that face. It is readily apparent that a finite edgeset must form a closed circuit of edges. We therefore call a finite edgeset a *loop*.

We will often refer to the planar dual of a polyhedron and, in the remainder of this section, if $\mathcal{H}$ is a polyhedron, we denote its planar dual by $\mathcal{H}^*$ and we might simply refer to it as the *dual*. Since we will often refer to a polyhedron and its dual simultaneously, it will be convenient to use the terms *node*, *arc*, and *cell* to refer to the dual versions of vertices, edges, and faces. Let $\mathcal{H}$ denote a polyhedron. A node in $\mathcal{H}^*$ corresponds to a face in $\mathcal{H}$, an arc in $\mathcal{H}^*$ joins nodes and occurs whenever two faces in $\mathcal{H}$ are adjacent. Although we already use the term "cell" when referring to the cells of a cellular automaton, this should not be a cause for confusion; a cell in $\mathcal{H}^*$ corresponds to a vertex in $\mathcal{H}$ and a vertex represents the location of a cell in a cellular automaton. Furthermore, we do not mention cellular automata anywhere else in this section. In the context of the dual of a polyhedron, a cell is a region bounded by a set of arcs just as a face is bounded by a set of edges in the original polyhedron. Just as $V(\mathcal{H})$ and $E(\mathcal{H})$ refer to the set of vertices and edges in a polyhedron, we write $\mathsf{nodes}(\mathcal{H}^*)$ and $\mathsf{arcs}(\mathcal{H}^*)$ to refer to the set of nodes and arcs in the dual. When $\mathcal{H}^*$ is the dual of an even-face-degree polyhedron $\mathcal{H}$, we have the following definition which corresponds to an edgeset in $\mathcal{H}$.

Definition 15 (e-curve)**.** Let $\mathcal{H}$ be an even-face-degree polyhedron, and $\mathcal{H}^*$ be its dual. For every edgeset in $\mathcal{H}$, there is an associated set of edges in $\mathcal{H}^*$ called an *e-curve*. According to the construction of the dual (see [7], Section 4.5), every edge in $\mathcal{H}$ is intersected by exactly one arc in $\mathcal{H}^*$ and all such intersections are transversal. An e-curve in $\mathcal{H}^*$ is a union of the set of edges that intersect edges in an edgeset in $\mathcal{H}$ and these arcs (or their nodes taken in sequence) naturally define a path.

Since e-curves are fully defined once a polyhedron is defined, and since e-curves are defined in the same plane as the polyhedron, we will often refer to the e-curves of a polyhedron without explicitly mentioning the dual of the polyhedron. Figures A.1 and A.2 illustrate e-curves in various polyhedra.

As a consequence of the way e-curves are derived from their edgeset counterparts, it is apparent that e-curves intersect themselves and each other only transversally. We can directly translate, to e-curves, the definitions given previously for edgesets that either intersect, self-intersect, or loop. In addition, it will be convenient to speak of a path crossing an e-curve. Let $\mathcal{H}$ be an even-face-degree polyhedron and $\mathcal{H}^*$ be its dual. A path P in $\mathcal{H}$ *crosses* e-curve C in $\mathcal{H}^*$ if an edge of P intersects an edge in C. If a and b are vertices in $\mathcal{H}$, then C is a *separator* of a and b if every path from a to b crosses C. Otherwise C is a *non-separator* of a and b.

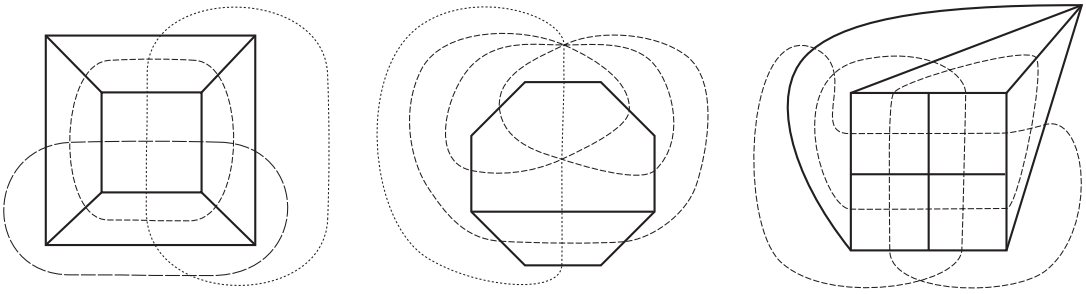

Figure A.1: Three even-face-degree polyhedra. The solid lines define polyhedral edges while the dashed lines indicate e-curves associated with edgesets. From left to right, the polyhedra have three, two and one edgesets respectively. Since the polyhedra are finite, all edgesets form loops.

Definition 16 (edgeset-coloring)**.** An *edgeset-coloring* is an assignment of colors to each edgeset subject to the restriction that any two edgesets that share a common vertex in $\mathcal{H}$ have different colors. If the number of colors required is finite, then $\mathcal{H}$ is said to have a *finite edgeset-coloring*.

Notice that an edgeset-coloring is an addressing-scheme. When we assign each e-curve the same color as its associated edgeset, we have a definition for coloring e-curves:

Definition 17 (e-curve-coloring)**.** An *e-curve-coloring* is an assignment of colors to e-curves such that by coloring each e-curve's associated edgeset, an edgeset-coloring is obtained.

Any e-curve-coloring results in a dual polyhedron with the property that for every cell, the set of e-curves bounding the cell all have different colors. Conversely, if the set of e-curves in the dual of a polyhedron can be colored such that this property is satisfied, then the polyhedron has an edgeset-coloring with the same colors.

Definition 18 (kink-free polyhedron)**.** A *kink-free polyhedron* is an even-face-degree polyhedron in which every pair of e-curves intersect at most once, and every e-curve does not self-intersect.

We would like to characterize the distance between two vertices by the e-curves which separate them. Figure A.2 shows an example of a non-kink-free polyhedron on the left and a kink-free polyhedron on the right. Notice that the shortest path between α and β in the non-kink-free polyhedron involves a non-separating e-curve, while for the kink-free polyhedron, only separating e-curves cross. We show that distances in kink-free polyhedra are characterized by the number of separating e-curves in the following theorem.

Lemma 39. *All e-curves in a kink-free polyhedron are infinite.*

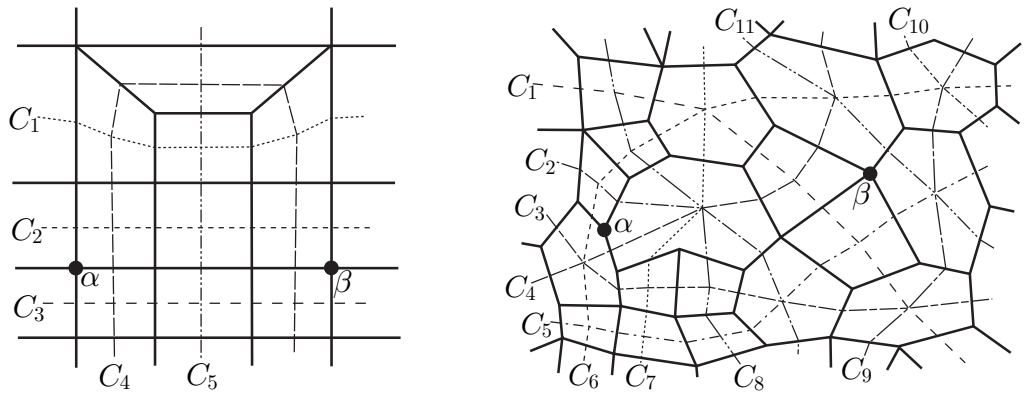

Figure A.2: These two figures show portions of infinite polyhedra (solid lines) and their e-curves (broken lines labeled C_i). The left polyhedron is not kink-free, and the shortest path from α to β crosses C_4 twice. The right polyhedron is kink-free, and all shortest paths from α to β cross the same set of e-curves exactly once. The three shortest-path e-curve sequences are $(C_2, C_8, C_7, C_1, C_4)$, $(C_2, C_8, C_7, C_4, C_1)$, and $(C_4, C_7, C_8, C_2, C_1)$.

Proof. An immediate consequence of the definitions for e-curves and kink-free polyhedron is that any e-curve that is not infinite is formed from a simple closed-loop of arcs such that no two arcs intersect one another. Assume for the sake of a contradiction that C_0 is a finite e-curve, and thus we can speak of the interior and exterior of C_0 (recall that polyhedra have no vertex accumulation points). Now C_0 contains at least one arc, and every arc is intersected by an edge (its unique dual edge), call it e_0 (as illustrated in Figure A.3). One vertex, v_0 of e_0 must be in the interior C_0. By the definition of an even-face-degree polyhedron, every vertex is the corner of an even-degree face of degree ≥ 4. Call this face F_0. The e-curve C_0 bisects F_0, and since F_0 has at least 4 sides, there is an edge, call it e_1, on F_0 adjacent to e_0 and in the exterior of C_0. Through this edge there must be an e-curve C_1 and this e-curve must enter the interior of C_0, since all e-curves intersecting a face meet at a common node and intersect each other transversally. By the definition of kink-free, self-intersection is forbidden, so C_0 and C_1 are distinct e-curves. But C_1 must again exit the interior of C_0 since the interior of C_0 is finite. Thus, C_1 intersects C_0 twice, a contradiction. □

Recall that our goal in this is to show that a particular class of polyhedra have finite shortest-path-invariant addressing-schemes. Our strategy is to show that a subset of kink-free polyhedra have finite e-curve-colorings, and to then use this property to prove the existence of our desired addressing-schemes.

Suppose we demonstrate that a particular kink-free polyhedron has a finite e-curve-coloring. The finite e-curve-coloring provides a finite addressing-scheme, but we cannot yet make the claim that it is also shortest-path-invariant. We must first show that: (1) every shortest path between two vertices crosses all separating e-curves exactly once (fairly obvious), and (2) all shortest paths between two vertices that cross non-separating e-curves cross the same number e-curves of

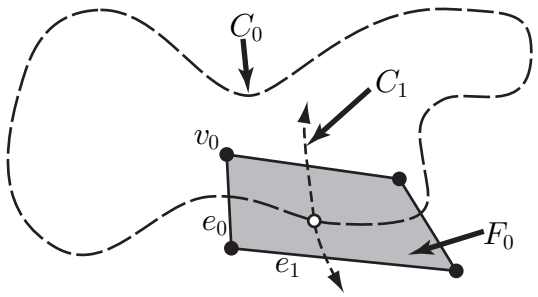

Figure A.3: Finite e-curves are forbidden in kink-free polyhedra. E-curves are represented in dashed lines, edges are solid lines, vertices are black dots, nodes are white dots, and face F_0, which is of degree ≥ 4, is shaded.

each color (not so obvious). We address these concerns in Lemma 41 by proving that for kink-free polyhedra, no non-separating e-curves are ever crossed on a shortest path between vertices.

In our upcoming discussion, it will be convenient to view the plane as mapped to the unit disc where the origin (chosen arbitrarily) is at the center and the unit circle is the boundary at infinity. Note that this boundary is not a point, and therefore two infinite lines may, or may not, intersect at infinity [2]. By Definition 15 and Lemma 39, every e-curve in a kink-free polyhedron is a doubly infinite sequence of distinct arcs and therefore all e-curves intersect the circle at infinity in two places. Every e-curve therefore bisects the unit disc.

We will at times find it useful to introduce the interior of a cell as a hole at the origin of the unit disc, thereby transforming the disc into an annulus. Anything exterior to the hole is defined to be "below" the hole. Think of the annulus as a cylinder with a hole at the top, and the circle at infinity at the bottom. Since all e-curves bisect the plane, and since we always chose the hole so that it does not intersect an e-curve, phrases such as "the half of the plane containing the hole" are well defined when it is understood that the two "halves" of the plane are with respect to a bisecting e-curve.

Let A be an e-curve. A point not on A is *above* A if it is in the half of the plane containing the hole, and it is *below* C if it is in the other half. Let B be an e-curve distinct from A. If A does not intersect B, then there are three topological possibilities: (1) If B cannot be continuously transformed so as to be arbitrarily close to the hole without any part of B passing through A, then B is said to be *below* A and we write $B \succ A$. (2) If the previous statement is true when the labels A and B are reversed, then B is said to be *above* A and we write $B \prec A$. (3) If neither of the previous two statements hold, then A and B are said to be *peers*.

When speaking of a kink-free polyhedron, we assume that the underlying

[2]This phenomenon corresponds to the distinction between parallel and ultra-parallel lines in hyperbolic geometry.

polyhedron and its dual are defined. We also assume that a cell is chosen (called the *origin cell*) and we let the interior of this cell be the hole of the annulus.

Lemma 40. *Let $\mathcal{H}$ be a kink-free polyhedron embedded in an annulus with origin cell F_0 serving as the hole. Then every point in the annulus is below at least one of the e-curves bounding F_0.*

Proof. Assuming that we label e-curves in a counter-clockwise order, let $C_0, \ldots, C_{m-1}$ be e-curves bounding origin cell F_0 as shown in Figure A.4. For $0 \leq k \leq m-1$, e-curve $C_{k \pmod m}$ intersects $C_{k-1 \pmod m}$ and $C_{k+1 \pmod m}$ (assume "$i \pmod m$" means the least non-negative integer k s.t. $i - k \equiv 0 \pmod m$). Since kink-free e-curves are forbidden from intersecting each other twice, we see that the union of all the regions below e-curves that bound the origin face cover the entire annulus—each e-curve has two arms extending away from F_0 and one arm from each pair of intersecting e-curves interlock. □

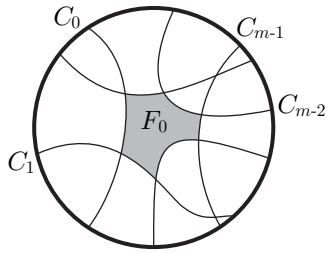

Figure A.4: An annulus with cell F_0 for the hole formed from e-curves $C_0, \ldots, C_{m-1}$.

Lemma 41. *Let $\mathcal{H}$ be a kink-free polyhedron and let α and β be any vertices in $\mathcal{H}$. Any shortest path from α to β crosses only separating e-curves, and every separating e-curve is crossed exactly once.*

Proof. Let α and β be two vertices of $\mathcal{H}$. By the definitions of separating and non-separating e-curves, and by a simple parity argument, any path from α to β must cross all separating e-curves an odd number of times and all non-separating e-curves an even number of times. Since all separating e-curves are crossed at least once, any shortest path from α to β: (1) crosses every separating e-curve, (2) never crosses any non-separating e-curve, and (3) never crosses a separating e-curve more than once. Consider the following algorithm for finding a path between α and β. We described a path from α to β by a list of e-curves that the path crosses starting from α. Assume that the list is initially empty and the current cell is the cell containing α.

Repeat the following procedure "Path-Find":

1. If the current cell contains β, then return the list and exit the algorithm.

2. Choose any separating e-curve bounding the current cell that has not already been crossed. Add this e-curve to the list and cross into the opposite cell.

We show that at every repetition, the following invariant is maintained: If the current cell does not contain the destination, then the current cell is bounded by at least one separating e-curve that has not already been crossed.

The invariant is proved by induction: we assume the invariant has held for all previous steps, and show this implies that it holds at the current step. The "current cell" contains the vertex where the end of the current path from a is located. It will be useful to view $\mathcal{H}$ embedded in an annulus (as described previously). We call the origin cell F_0, and define it to be the current cell. At initialization, the invariant is trivially true. If the current cell contains the destination cell, then the invariant also holds. Therefore, assume that the current cell contains neither α nor β.

For the sake of a contradiction, suppose that the invariant is false for the current cell. This implies that every e-curve bounding F_0 is either a separating e-curve the algorithm previously crossed, or a non-separating e-curve. Consider any e-curve C bounding F_0. If C is a separator, then a is below C, since C must pass between F_0 and α exactly once. This implies that β is above C since C separates α and β. If C is a non-separator, then α is above C since the current path has not crossed C. In this case, β is also above C since C is a non-separator. Since C was chosen arbitrarily from among e-curves bounding F_0, there no e-curve bounding F_0 for which β is below. The two cases for C are illustrated in Figure A.5. By

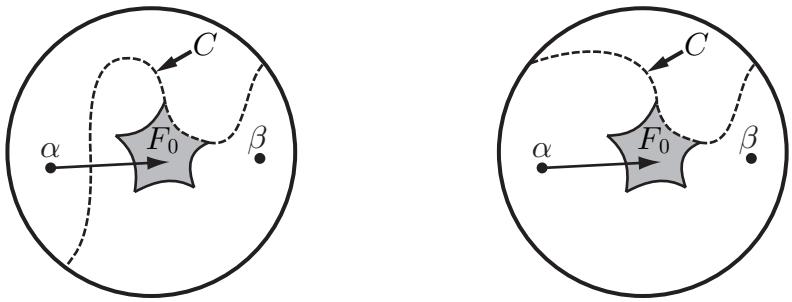

Figure A.5: From left to right, these diagrams illustrate the configuration a separating and a non-separating for an e-curve C which bounds the origin cell F_0, according to the construction in Lemma 41.

Lemma 40, every vertex outside of F_0 is below at least one e-curve bounding F_0. This is a contradiction, and therefore it must be the case that the invariant also holds for the current step. □

With Lemmas 39 and 41 proved, it only remains to show that kink-free polyhedra (with minimal vertex degree 4) have finite e-curve-colorings. Unfortunately, Theorem 53 and Conjecture 2 suggest that these restrictions are not sufficient criteria for the set of e-curves to have a finite coloring. However, we can add further

restrictions to the set of kink-free polyhedra without eliminating any polyhedra in which we are ultimately interested.

Definition 19 (nice-e-curve-set)**.** Let $\mathcal{H}$ be a kink-free polyhedron and let $\mathcal{S}$ be its set of e-curves. $\mathcal{S}$ is called a *nice-e-curve-set* if the maximum degree of any cell is p, and if one of the following two statements is true:

1. All cells have degree at least 5, and at least 2 e-curves intersect at every node.

2. All cells have degree at least 4, and at least 3 e-curves intersect at every node.

Definition 20 (level of a node and e-curve)**.** Let $\mathcal{H}$ be a polyhedron with a nice-e-curve-set $\mathcal{S}$ and origin cell F_0. The *level* $l(a)$ of a node $a \in V(\mathcal{H}^*)$, is defined

$$l(a) = \mathsf{dist}(\mathsf{nodes}(F_0), a).$$

The *level* $l(C)$, of an e-curve $C \in \mathcal{S}$ is defined

$$l(C) = \min_{a \in \mathsf{nodes}(C)} l(a).$$

By this definition, any e-curve with a node on the boundary of F_0 is at level 0.

Lemma 42. *Let G be a graph with $S \subseteq V(G)$, vertices $a, b \in V(G)$, and path P from a to b. Then*

$$|\mathsf{dist}(S, a) - \mathsf{dist}(S, b)| \leq |P|.$$

Proof. Since function dist is a metric on the set of vertices, by the triangle inequality and by the definition of dist between a set and a vertex, we have:

$$\begin{aligned} \mathsf{dist}(S, a) &\leq \mathsf{dist}(S, b) + \mathsf{dist}(a, b), \quad \text{and} \\ \mathsf{dist}(S, b) &\leq \mathsf{dist}(S, a) + \mathsf{dist}(a, b). \end{aligned}$$

Since $\mathsf{dist}(a, b) \leq |P|$, the theorem follows. □

Paths (which include finite paths, rays and double-rays) were previously defined in Section 3.1, but we describe a few additional notational conventions that we use in the remainder of the Appendix. For convenience, we will sometimes denote a path by an ordered list of its vertices such as $v_0 v_1 \dots v_k$, $v_0 v_1 \dots$, or $\dots v_{-1} v_0 v_1 \dots$ if the path is either finite, a ray or a double-ray respectively. If $P = v_0 v_1 \dots v_k$ is a finite path then Pw denotes the path $v_0 v_1 \dots v_k w$ or if $P = v_0 v_1 \dots$ is a ray then wP denotes the ray $w v_0 v_1 \dots$. Let $P = v_0 v_1 v_2 \dots$ be a ray. By removing some finite number of vertices from the beginning of a P, we get various subrays such as $P_1 = v_1 v_2 \dots$ or $P_k = v_k v_{k+1} \dots$.

Lemma 43. *Let G be a graph with $S \subseteq V(G)$ and $P = x_0, x_1, \ldots, x_n$ any shortest path from S to x_n. Then for each $0 \leq k \leq n$, the subpath Px_k is a shortest path from S to x_k and $|Px_k| = k$.*

Proof. By definition, $|Px_k| = k$ for all $0 \leq k \leq n$, and in particular, $|P| = |Px_n| = n$. For the sake of a contradiction, suppose that Px_{n-1} is not a shortest path from S to x_{n-1}. Then a path Q from S to x_{n-1} exists such that $|Q| \leq n-2$. But then path Q augmented with edge $x_{n-1}x_n$ is also a path from S to x_n, and it has length at most $|Q|+1 \leq n-1 < n$, a contradiction. Therefore Px_{n-1} is a shortest path from S to x_n. By induction, the argument extends to $k \in \{0, \ldots, n-2\}$ as well. □

Lemma 44. *Let $\mathcal{S}$ be a nice-e-curve-set with origin cell F_0, and let $A, B \in \mathcal{S}$. If $B \succ A$, then $l(B) > l(A)$.*

Proof. Any shortest path from F_0 to a node in B, first passes through a node in A. By Lemma 43, there is always a node in A with a smaller level than any node in B. □

We do not use the term "polygon" in the main body of the thesis, and so we use it here with the following meaning. All sets of e-curves under consideration are assumed to be kink-free so that any two e-curves intersect at most once.

Definition 21 (polygon)**.** Any set P of mutually intersecting e-curves that enclose a finite connected region of the plane are said to form a *polygon*. We require that a polygon has the following property. If a and b are two points within the enclosed-region, and not on an e-curve in P, then a and b can be connected by a curve that does not intersect e-curves in P. Finally, every e-curve in P contributes more than one point to the boundary of the region (our polygon roughly corresponds to the usual meaning of a "simple polygon").

We call an intersection node between two e-curves in a polygon a *polygon-corner* if this node is on the boundary of the finite region defined by the polygon. The segment of an e-curve between two corner-nodes is called a *polygon-segment*. It can be reasoned that there is a one-to-one correspondence between polygon-segments, polygon-corners, and e-curves in a polygon. The *degree of the polygon* is the cardinality of the set of its set of e-curves. Therefore, every set of e-curves defining a cell forms a polygon, and any two cells that share a common arc can be combined to create a larger polygon (the e-curve corresponding to the common arc is not part of the new polygon).

Note that if an e-curve C bisects a polygon P_1, then two new polygons P_2 and P_3 are formed by two different subsets of P_1 and the e-curve C. The degree of P_2 (or P_3) no more than that of P_1. If C intersects a polygon-corner of P_1, then at least one of P_2 and P_3 is of degree strictly less than P_1.

Definition 22 (triangle)**.** In an even-face-degree polyhedron, a set of three pairwise intersecting e-curves are said to form a *triangle*. An even-degree polyhedron with no triangles is called *triangle-free*.

Lemma 45. *Let $\mathcal{H}$ be an even-face-degree polyhedron where all vertices have degree at least four. Then P is triangle-free.*

Proof. For the sake of a contradiction, suppose there exists a set of three e-curves P that form a triangle. P cannot define a cell, because the dual of a degree-three cell is a degree-three vertex. Therefore, some other e-curve bisects P and forms two new degree-three polygons P_1 and P_2. By Definitions 5 and 12, polyhedra are vertex-accumulation-point-free, and therefore the process of bisecting triangles ends after a finite number of bisections, since a eventually a polygon that is also a cell is reached. This final polygon corresponds to a degree-three cell, which is a contradiction. □

Definition 23 (square-and-square)**.** Let $\mathcal{S}$ be nice-curve-set. A *square-and-square* structure $S = P_1 \cup P_2$ is a set of e-curves in $\mathcal{S}$ for which P_1 and P_2 are both degree-four ("square") polygons. Furthermore P_1 and P_2 share a common polygon-segment corresponding to e-curve C such that $(P_1 \cup P_2) \setminus \{C\}$ is a polygon of degree either four or five. See the first two diagrams in Figure A.6 for examples.

Definition 24 (square-and-pentagon)**.** Let $\mathcal{S}$ be a nice-curve-set. A *square-and-pentagon* structure $S = P_1 \cup P_2$ is a set of e-curves in $\mathcal{S}$ for which P_1 is a degree-four polygon, and P_2 is a degree-five ("pentagon") polygon. Furthermore P_1 and P_2 share a common polygon-segment corresponding to e-curve C such that $(P_1 \cup P_2) \setminus \{C\}$ is a polygon of degree either five or six. See the last two diagrams in Figure A.6 for examples.

Lemma 46. *A nice-e-curve-set contains neither square-and-pentagon nor square-and-square structures.*

Proof. Let $S = P_1 \cup P_2 \cup C$ be either a square-and-square structure, or a square-and-pentagon structure with P_1, P_2 and C as described in Definitions 23 and 24 respectively. For the sake of a contradiction, suppose that a nice-e-curve-set $\mathcal{S}$ contains such a S. The existence of a degree-four polygon in S implies the existence of a degree-four cell in S and so, by Definition 19, at least three e-curves intersect at every node in $\mathcal{S}$. It can be shown with a few cases, that e-curve C passes through at most one the corner-nodes in polygon $(P_1 \cup P_2) \setminus C$. Therefore, at least one of the two nodes that C passes through in S has only two e-curves passing through it (when just the e-curves in S are considered). Call this node c. But each node has at least three e-curves passing through it and so, there exists

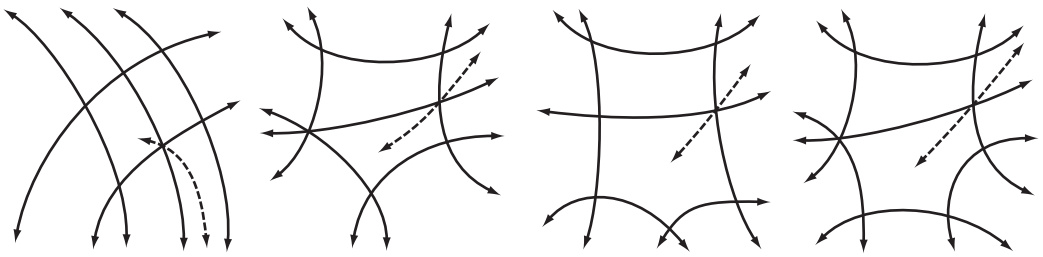

Figure A.6: The solid lines in each of the four figures represent sets of e-curves that form either the square-and-square, or the square-and-pentagon structures. In an nice-e-curve-set, these structures are forbidden since the their existence implies the e-curve shown as a dashed line, and this e-curve implies (eventually) the existence of triangle polygons. But by Lemma 45, nice-e-curve-sets are triangle-free. Lemma 46 provides the proof in full.

an e-curve $D \in \mathcal{S} \setminus S$ that also passes through c. In Figure A.6, D is represented by the dashed line.

Suppose that S is a square-and-square structure. Then e-curve D bisects one of the two degree-four polygons, say polygon P_1, at a corner-node of P_1. But then a subset of $P_1 \cup D$ forms a triangle. By Lemma 45, nice-e-curve-sets are triangle-free, and we have a contradiction. Therefore square-and-square structures are forbidden in $\mathcal{S}$.

Suppose that S is a square-and-pentagon structure. Then e-curve D must bisect P_2 (the pentagon) to avoid triangles. Assuming that D is also avoids creating a triangle with a subset of P_2, then $P_2 \cup D$ creates a square-and-square structure where D plays the role that C took previously. By the previous paragraph, this leads to a contradiction. Therefore square-and-pentagon structures are forbidden in $\mathcal{S}$. □

Lemma 47. *Let A, B, and C be e-curves in a nice-e-curve-set such that $B \succ A$. If C intersects B at node β, and C intersects A at node α, then $l(\beta) > l(\alpha)$.*

Proof. Let F_0 be the origin face. For the sake of a contradiction, assume that $l(\beta) \leq l(\alpha)$. Let $P\beta$ be a shortest path from F_0 to β. Let γ be the node on $P\beta$ that is the final node on curve A (as the path is traversed from F_0 towards β). Observe that $\gamma \neq \alpha$, for otherwise $l(\beta) > l(\alpha)$ (by Lemma 43) which contradicts the assumption $l(\beta) \leq l(\alpha)$. Let $\gamma A \alpha$ be the path from γ to α along curve A. By Lemma 42, and by the fact that $l(\alpha) \leq l(\gamma)$, we have $l(\alpha) - l(\gamma) \leq |\gamma A \alpha|$. Let φ be the set of nodes along A, between γ and α. Then $|\varphi| \geq (l(\alpha) - l(\gamma) + 1) - 2$. Let γ' be the first node after γ on path $\gamma P \beta$. Node γ' is distinct from β (since triangles are forbidden by Lemma 45 and e-curves C and A already form two sides of a triangle), and furthermore, no e-curve intersects both γ' and any node of φ (again because triangles are forbidden). Let ρ be the set of nodes along path $\gamma P \beta$,

between nodes γ' and β. Then $|\rho| = (l(\beta) - l(\gamma) + 1) - 3 \leq (l(\alpha) - l(\gamma) + 1) - 3$. Figure A.7 illustrates the sets of nodes φ and ρ. Every e-curve through a node

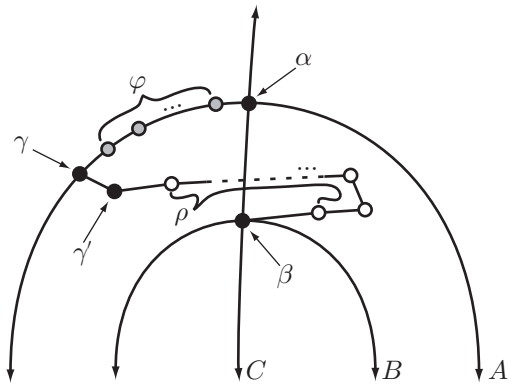

Figure A.7: Diagram illustrating sets of nodes φ and ρ.

in φ, must pass through a distinct node in ρ, since otherwise a forbidden triangle would be formed either between that e-curve and A and C, or between A and two e-curves passing through nodes in φ that pass through a common node in ρ. But $|\varphi| - |\rho| \geq ((l(\alpha) - l(\gamma) + 1) - 2) - ((l(\alpha) - l(\gamma) + 1) - 3) = 1$. Therefore, the assumption that $l(\beta) \leq l(\alpha)$ leads to a contradiction. □

Theorem 48. *Let $\mathcal{S}$ be a nice-e-curve-set with faces of maximum degree p. Then the e-curves in $\mathcal{S}$ can be colored with at most $5p-6$ colors such that the e-curves bounding any cell are all of different colors.*

Proof. Our strategy is to provide an algorithm for coloring all e-curves in a particular order with the property that whenever an e-curve is to be colored, there are at most $5p - 7$ already-colored e-curves constraining the choice of color.

As we have done previously, we map the set of e-curves (along with all components of the defining polyhedron) onto the annulus. The algorithm begins by choosing any cell to be the origin cell. We denote this cell F_0 and recall that the interior of this cell serves as the hole of the annulus. The set of m e-curves bounding F_0 are denoted $C_0, \ldots, C_{m-1}$. Since every cell has degree at most p, the bounding e-curves can be colored with at most p colors.

For illustrative purposes, it will at times be convenient to display the annulus as a flattened cylinder as shown in Figure A.8.

The algorithm colors e-curves in $\mathcal{S}$ one level at a time, and one node at a time, starting with level zero and increasing. The e-curves $C_0, \ldots, C_{m-1}$ are colored with $\leq p$ colors. Figure A.9 illustrates the constraints on a set of e-curves at a node at level zero. Consider any two neighboring e-curves C_j and C_k that bound F_0 and meet at node a. By the definition of a polyhedron, a finite number of e-curves can intersect at every node. The uncolored e-curves at a can be colored

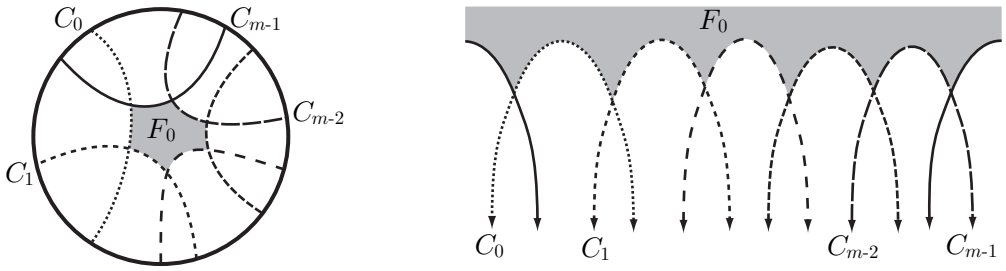

Figure A.8: The left-hand figure shows a set of m e-curves $C_0, \ldots, C_{m-1}$ displayed on the annulus. The hole is the cell F_0 is bounded by the m e-curves, and the circle represents the circle at infinity. The right-hand figure shows the e-curves and hole mapped onto a cylinder. The hole F_0 from the annulus is at the top of the cylinder, and the circle at infinity is the bottom of the cylinder. The left and right edges are connected.

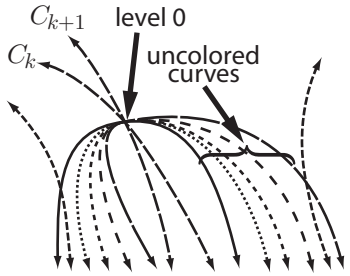

Figure A.9: Coloring a bundle of e-curves at level 0.

with two new colors by alternation. By Lemma 45, three pairwise intersecting e-curves are forbidden, and each bundle of uncolored e-curves can see at most two neighboring bundles. Therefore, the pairs of new colors used for each bundle can be alternated for bundles intersecting node around F_0. This requires at most three pairs of additional colors. Therefore, the level zero e-curves require at most $p + 6$ colors. Since $p \geq 4$, $p + 6 < 5p - 6$.

With all e-curves at level 0 colored, the algorithm chooses any level-one node, and colors all uncolored e-curves around that node. Then, a new level-one node is then chosen and the process repeated until all e-curves at level one are colored. The algorithms then proceeds to level two, and repeats. Since the degree of every node is finite, there are only a finite number of e-curves to be colored at each level. Therefore, every e-curve will eventually be colored. Figure A.10 illustrates some of this process for a subset of e-curves up to level two.

We now show that when an e-curve is currently being colored by the algorithm, it is not constrained by more than $5p - 7$ colors. To show this, our we first show that the e-curve currently being colored intersects at most four already-colored e-curves. So, for the sake of a contradiction, suppose that e-curve C currently being colored intersects five or more other e-curves. Since triangles are forbidden, all e-curves that C intersects are pairwise disjoint. An e-curve C is a double-ray

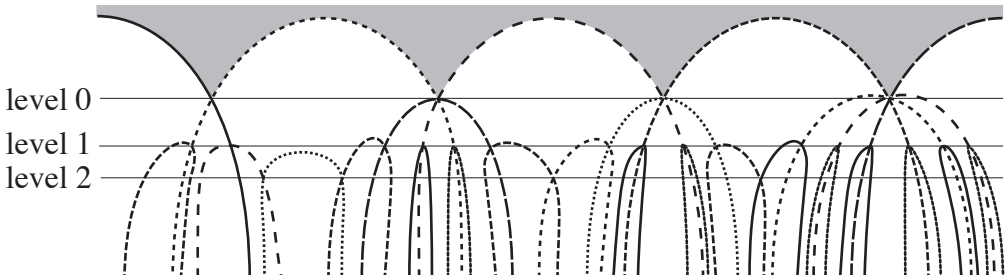

Figure A.10: A subset of e-curves in $\mathcal{S}$ where cell-degree ≥ 5. Each node and e-curve has a well-defined level.

and therefore intersects the circle- at-infinity in two places, call them *tail-1* and *tail-2*. Any e-curve intersecting C separates either tail-1 or tail-2 from F_0, the hole of the annulus. When a set of pairwise non-intersecting e-curves separate a common tail (or any common point on the annulus) from the hole, then the relation $\succ$ is a total order on the set. Therefore, if a set of five pairwise disjoint e-curves intersect C, then at least three of these e-curves A_1, A_2, and A_3 separate a common tail of C. Let the labels on the A_i be such that $A_3 \succ A_2 \succ A_1$. By Lemma 44, $l(A_3) > l(A_2) > l(A_1)$, and by the order in which e-curves are colored, $l(C) \geq l(A_3)$. Let a_1, a_2 and a_3 be the nodes along curve C that also belong to A_1, A_2 and A_3 respectively. By the fact that triangles are forbidden (Lemma 45), we may suppose that A_1, A_2 and A_3 are such that a_1a_2 and a_2a_3 are arc segments along curve C, and therefore, by Lemma 47, $l(a_3) = l(a_2)+1$ and $l(a_2) = l(a_1)+1$. For example, if there were a node a' between a_1 and a_2 along C, then we could choose the curve that intersects a' rather than A_1.

Let $A = A_1$, $B = A_2$, $c = a_1$, and $c' = a_2$. Let $b \in \mathsf{nodes}(B)$ be such that $l(B) = l(b)$ and let Pb be a shortest path from F_0 to b. Let a be the last node (starting from F_0) along path Pb to also be on curve A. If $a = c$, then $l(c) < l(b)$ (by Lemma 43), and so $l(C) \leq l(c) < l(b) \leq l(B)$, which is impossible since B is colored before C. Therefore $a \neq c$. If $b = c'$ then $l(c) < l(c')$, and $l(C) = l(c') < l(c) \leq l(C)$ which is a contradiction. Therefore $b \neq c'$. Let φ be the set of nodes between c and a on curve A. By Lemma 42, $|\varphi| \geq (l(c)-l(a)+1)-2$. We wish to prove that either ab is an arc, or some e-curve passing through a node in φ also passes through a node on B between c' and b. If this happens, there would exist a curve C' that intersect B and A and is distinct from C. For the sake of a contradiction, suppose that ab is not an arc and that no e-curve passes through a node in φ and a node on B between c' and b. Then there exists node a' on path aPb, distinct from a and b such that aa' is an arc. Since triangles are forbidden, an e-curve passing through a node in φ cannot also pass through a' from φ. Let ρ be the set of nodes on the path aPb between a' and b. The current setup is illustrated in Figure A.11. Since e-curve C is colored after e-curves A and B, and since $l(b) = l(B)$, it must be true that $l(b) \leq l(c)$. By this inequality and by Lemma 43, $|\rho| \leq (l(b)-l(a)+1)-3 \leq (l(c)-l(a)+1)-3$. Since triangles are

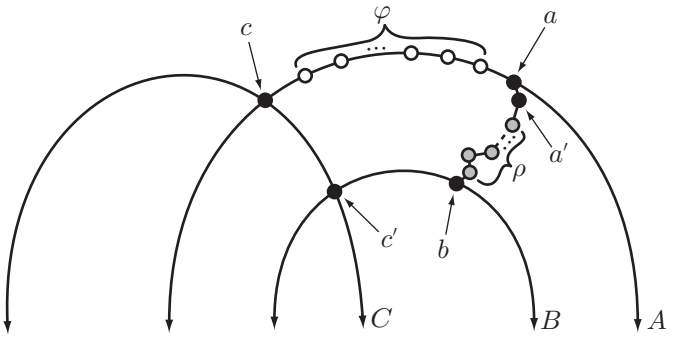

Figure A.11: The e-curves passing through nodes in φ must also pass through nodes in ρ.

forbidden, every e-curve passing through a node in φ must also pass through a node in ρ. However, $|\varphi|-|\rho| \geq ((l(c)-l(a)+1)-2)-(3 \leq (l(c)-l(a)+1)-3) \geq 1$, and there must exist an e-curve $C' \neq C$ passing through e-curves A_1 and A_2.

We can repeat the same argument as above if we let $A = A_1$, $B = A_2$, $a_1 = c$, and $a_2 = c'$. Therefore there exists $C'' \neq C$ which also intersects e-curves A_2 and A_3. Figure A.12 illustrates the one of a set of a potential relative-positions for e-curves A_1, A_2, A_3, C, C', and C''. If $C' = C''$ then this e-curve intersects A_1, A_2,

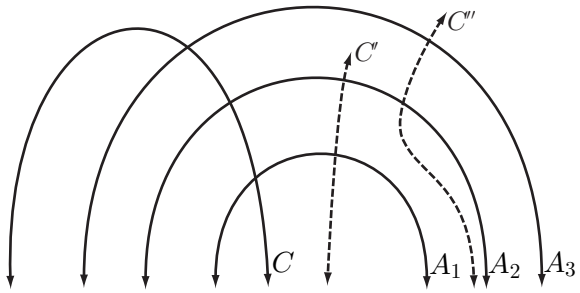

Figure A.12: The middle e-curve must create a forbidden structure.

and A_3, and one of the two square-and-square structures is created. If $C' \neq C''$, either at least one of C' and C'' intersects A_1, A_2 and A_3 (creating one square-and-square structure), or C' and C'' intersect at e-curve A_2 (creating a different square-and-square structure), or C' and C'' intersect at a common node between A_1 and A_2 or between A_2 and A_3 (creating a square-and-pentagon structure). By Lemma 46, square-and-square and square-and-pentagon structures are forbidden for nice-e-curve-sets. Therefore an e-curve, currently being colored, intersects at most four previously-colored e-curves.

By the order in which e-curves are colored, there cannot exist an already colored e-curve D below C. Let $a_1, \ldots, a_4$ be the (at most) four nodes (in order) where C intersects already colored e-curves. We must argue that there can be

no not-already-colored e-curves intersecting C between each of the e-curve segments $\{a_1, a_2\}$, $\{a_2, a_3\}$, and $\{a_3, a_4\}$. Each of the four already-colored e-curves intersecting C have levels $\leq l(C)$ since they were colored before C. However, if an e-curve D intersects C at one of the aforementioned segments, then it is below at least one already colored e-curve, say E. By Lemma 47, this implies that $l(E) < l(D)$. But since E is not yet colored, $l(E) \geq l(C) \geq l(E)$. This is a contradiction. Therefore, the arcs a_1a_2, a_2a_3, and a_3a_4 exist. These three arcs are part of three different cells with degree at most p. Therefore, C might be constrained by $p-1$ already-colored faces in each of these. Below a_1 and a_4, C can be constrained to most $p-2$ already-colored e-curves each. This brings the total to $3(p-1)+2(p-2) = 5p-7$ constraints at each step. Therefore, at most $5p-6$ colors are needed. □

Theorem 49. *Every polyhedron G in $\langle 4, 4\rangle \cap \mathcal{P}_{\text{evn,fnt}}$ is kink-free.*

Proof. The restriction $\mathcal{P}_{\text{evn,fnt}}$ implies that G is an even-face-degree polyhedron. Therefore, edgesets (and their associated e-curves) are well defined. By Definition 18, it only remains to show that no edgeset intersects itself, and no pair of edgesets intersect each other at two distinct faces. For the sake of a contradiction, suppose that either a single e-curve bounds a finite region by intersecting itself at a single face, or that two e-curves bound a finite region by intersecting each other at two distinct faces. Let C be a set of one or two faces where the e-curve(s) intersect(s) to bound a finite region of faces. Let B be the set of faces that the e-curve(s) pass(es) through to form the boundary, excluding the intersection face(s). Let A be the set of faces bounded by, but not including the sets B and C. Let P be the polyhedron defined by faces A. Figure A.13 illustrates P, A, B, and C. For the

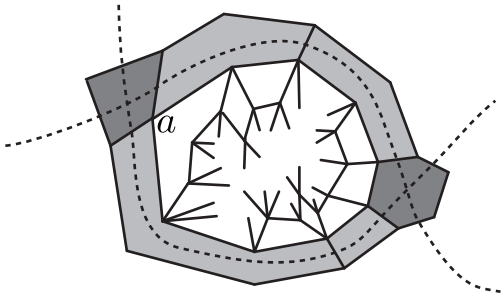

Figure A.13: Two bounding e-curves are shown as dashed lines. The faces in set C are darkly shaded, while the faces in set B are lightly shaded. Set A consists of the faces bounded by sets B and C. Polyhedron P is defined by the faces of A.

remainder of this proof, we imagine that P is in isolation from B and C. We still reference B and C, but when speaking of vertices and edges in P, we only count edges and vertices that make of faces of A. The boundary of P consists of the set of vertices and edges that are common to A and to $B \cup C$. Therefore, some

of the vertices on the boundary of P have degree one or two less then they have when considered the graph G. For example, in Figure A.13, the vertex labeled a had degree two in P and degree four in G. We call the large face formed by the boundary of P the exterior face ∂. In the calculations for the proof, we use the following conventions. If F is a face, then $V(F)$ is its set of vertices and $E(F)$ is its set of edges. The degree of a face F is denoted $d(F)$, and the degree of a vertex $v \in V(F)$ is denoted $d(v)$. For any set of faces A, we let $V(A)$ and $E(A)$ denote the set of all vertices and edges (respectively) that form the faces of A.

Let $e = |E(P)|$, $v = |V(P)|$, and $f = |A| + 1$ where the "+1" is to account for external face of P. According to Euler's formula

$$f + v - e = 2. \tag{A.1}$$

Let b be the number edges (equivalently vertices) on the boundary. Since the degree of every face in A is least four, and since every edge is shared by two faces (when we include the external face), we have the inequality $2e \geq 4\,|A| + b$. Therefore

$$f \leq \frac{2e - b}{4} + 1. \tag{A.2}$$

Let d be the sum of the degrees of all vertices on the boundary. Since each edge has two end-vertices, we have the inequality $2e \geq 4\,|V(A) \setminus V(\partial)| + d$. The total number of vertices is $|V(A) \setminus V(\partial)| + b$, and therefore

$$v \leq \frac{2e - d}{4} + b. \tag{A.3}$$

Combining inequalities (A.2) and (A.3) with equation (A.1) gives the inequality

$$\frac{2e - b}{4} + 1 + \frac{2e - d}{4} + b - e \geq 2 \iff 3b - 4 \geq d. \tag{A.4}$$

Now the degree of every vertex in $V(\partial) \setminus V(C)$ is at least three. Only vertices in $V(\partial) \cap V(C)$ can have degree two or less. This can only occur when face in C has degree four, and there is at most one such vertex per degree-four face in C. An example is shown (notice vertex a) in Figure A.13 . Therefore $|C|$ is an upper-bound on the number of vertices with degree two or less. Therefore $d \geq 3(b - |C|) + 2\,|C| = 3b - |C|$. Substituting this lower bound on on d into A.4 gives

$$|C| \geq 4.$$

This contradicts the assumption that $1 \leq |C| \leq 2$ (and also proves the stronger result that $\langle 4, 4\rangle \cap \mathcal{P}_{\text{evn,fnt}}$ is triangle-free). Therefore all G in $\langle 4, 4\rangle \cap \mathcal{P}_{\text{evn,fnt}}$ are kink-free polyhedra. □

Lemma 50. *Let G be a graph in $\mathfrak{G} \cap \mathcal{P}_{\text{par}}$ with the strong descendant property. If $G \in \mathcal{P}_{\text{evn}}$ then there are no cousin-edges, and if $G \in \mathcal{P}_{\text{odd}}$ then every face has exactly one cousin-edge.*

Proof. By Lemmas 7 and 8, it is clear that for any $G \in \mathfrak{G} \cap \mathcal{P}_{\text{par}}$, every face has at most two cousin-edges. There is at most one edge whose end-vertices have the maximum shell distance for the face, and we call this the "top" cousin-edge. There is at most one edge whose end-vertices have the minimum shell distance for the face, and we call this the "bottom" cousin-edge. Furthermore, it is clear that every even-degree face has either exactly two cousin-edges or no cousin-edges, and every odd degree face has exactly one cousin-edge. Therefore, the theorem holds for every $G \in \mathcal{P}_{\text{odd}}$.

Suppose $G \in \mathcal{P}_{\text{evn}}$ and let F_1 be a face. Suppose that F_1 has a bottom cousin-edge. Then the bottom cousin-edge of F_1 is the top cousin-edge of another face F_2. Since F_2 has a top cousin-edge, it must also have a bottom cousin-edge. Notice that this creates an endless chain of faces. However, the level of each successive cousin-edge in this sequence is strictly decreasing, and will therefore eventually arrive at a face incident on the origin. But no face on the origin has a bottom cousin-edge, and therefore F_1 must have had no cousin-edges. Therefore there can be no cousin-edges in $G \in \mathcal{P}_{\text{evn}}$. □

Although we have restricted most of the discussion so far to polyhedra with even-degree faces, we can extend the definition and results concerning addressing-schemes to certain polyhedra having one or more odd-degree faces. We give the following definition which is designed to remove all cousin-edges from the a graph.

Definition 25 (shortest-path subgraph)**.** Let G be a simple connected graph and let $a^* \in V(G)$ be a fixed vertex called the origin. For $b \in V(G)$, let $\mathsf{dist}(b)$ denote the shortest-path distance to the origin. For all b, c such that $\{b, c\} \in E(G)$, either $\mathsf{dist}(b) = \mathsf{dist}(c)$, or $|\mathsf{dist}(b) - \mathsf{dist}(c)| = 1$. We define a subgraph $\mathsf{sp}(G, a^*)$ of G by removing every edge $\{b, c\} \in E(G)$ for which $\mathsf{dist}(b) = \mathsf{dist}(c)$. Graph $\mathsf{sp}(G, a^*)$ is called the *shortest-path subgraph* of G (centered at origin a^*).

Lemma 51. *For every simple connected graph G with $a^* \in V(G)$ the shortest-path subgraph $\mathsf{sp}(G, a^*)$ is connected with $V(G) = V(\mathsf{sp}(G, a^*))$, and has no odd cycles.*

Proof. Since the graph G is connected, there is a shortest path from the origin to every vertex. By Lemma 43, the distance between neighbor vertices along any shortest path is one. Therefore, every vertex in G, along with any shortest path subgraph of G, is also in the shortest-path subgraph.

The vertices of the shortest-path subgraph are partitioned into two sets according to the parity of their distance from the origin. Therefore, the shortest-path subgraph is bipartite, and thus contains no odd cycles. □

Proof. Let $\mathsf{sp}(G, a^*) = (V, E')$ be the shortest-path subgraph of graph $G = (V, E)$. Let a^*Pb be a shortest path from vertex a^* to b in G. Let $\{u, v\} \in E_{a^*Pb}$.

By definition $|\mathsf{dist}(u) - \mathsf{dist}(v)| = 1$ and therefore $\{u, v\} \in E'$. It follows that $\mathsf{sp}(G, a^*)$ is connected.

Let V_{evn} be the set of vertices with even shortest-path distance and V_{odd} be the set of vertices with odd shortest-path distance. By definition, $\{b, c\} \in E' \implies |\mathsf{dist}(b) - \mathsf{dist}(c)| = 1 \implies b \in V_{\text{evn}}$ and $c \in V_{\text{odd}}$ or $\implies c \in V_{\text{evn}}$ and $b \in V_{\text{odd}}$. Therefore $\mathsf{sp}(G, a^*)$ is bipartite and thus contains no odd cycles. □

Lemma 52. *Let P be a kink-free polyhedron with an e-curve-set whose e-curves can be colored with at most n colors such that e-curves bounding any cell (a face in the dual) of P receive different colors. Then P has a shortest-path-invariant addressing-scheme with n colors.*

Proof. Let a be the origin of P and let b be some other vertex. By Lemma 41, any shortest path from a to b crosses the same set of separating e-curves. A separating e-curve corresponds to an edgeset, so let every edge in the edgeset receive the same color as the separating e-curve. Since every edge corresponds to a separating e-curve, every edge in the graph receives one of n colors. Each of the edges around a face receive a distinct color because a cell formed by a set of separating e-curves in the dual graph P^* corresponds to a vertex with its set of incident edges P. □

The following is a restatement of Theorem 27, provided for convenience before we give its proof.

Theorem 27. *Every graph in $((\langle 3, 7\rangle \cup \langle 4, 5\rangle) \cap \mathcal{P}_{\text{fnt,par}}) \cup \langle \infty, 3\rangle) \cap \mathcal{Q}_{\text{bdd}}$ has a finite shortest-path-invariant addressing-scheme with respect to any cell chosen as the origin. The addressing-scheme requires at most $5q_{\max} - 6$ colors where $q_{\max}$ is the maximum vertex degree in the graph.*

Proof. If $G \in \langle \infty, 3\rangle$ then G is a tree and trivially only $q_{\max}$ colors are needed for the edge-coloring.

For the cases of $G \in \langle 3, 7\rangle \cup \langle 4, 5\rangle$, we note that cousin-edges are never included on a shortest path from the origin to a vertex, and therefore never contribute to the shortest-path-invariant address of a vertex.

Let $G \in \langle 3, 7\rangle \cap \mathcal{P}_{\text{fnt,par}}$. Arbitrarily choose $a^* \in V(G)$, and let $G' = \mathsf{sp}(G, a^*)$ be the shortest-path subgraph of G centered at origin a^*. By Lemma 50, the parity condition on face degrees ensure that each face has at most one bounding cousin-edge. By Lemma 8, $G \in \mathfrak{G}$ implies that each vertex is incident on at most two cousin-edges. Therefore $G' \in \langle 4, 5\rangle \cap \mathcal{P}_{\text{fnt,evn}}$ with maximum vertex degree $q'_{\max} = q_{\max}$ if the parity was even, and $q'_{\max} \leq q_{\max} - 1$ if the parity was odd. By Theorem 49, G' is kink-free and has the property that its e-curves intersect at least three to a node (in the dual), and all of its cells (in the dual) have degree at least four. Therefore G' is a nice-e-curve-set according to Definition 19. By Lemma 52,

G' has a shortest-path-invariant addressing-scheme with at most $5q'_{\max} - 6$ colors. Now color the non-cousin-edges of G according to their colors in G'. If the face parity of G is even, then $G = G'$ and we are done. If the face parity of G is odd, then $q'_{\max} \geq q_{\max} - 1$ and G' is colored with at most $5q_{\max} - 11$ colors. At most three extra colors are needed to color the cousin-edges of G, and therefore at most $5q_{\max} - 8$ colors are needed for the shortest-path-invariant addressing-scheme of G in this case.

Let $G \in \langle 4, 5\rangle \cap \mathcal{P}_{\text{fnt,par}}$. Arbitrarily choose $a^* \in V(G)$, and let $G' = \mathsf{sp}(G, a^*)$ be the shortest-path subgraph of G centered at origin a^*. If the faces of G are of even degree, then $G' = G$ and by Theorem 49, Definition 19, Theorem 48, and Lemma 52, G has a shortest-path-invariant addressing-scheme with at most $5q_{\max} - 6$ colors. If the faces of G' are odd, then $G \in \langle 5, 5\rangle$, and by Theorem 10 we see that $G' \in \langle 8, 4\rangle \cap \mathcal{P}_{\text{fnt,evn}}$ (we know that the degree of every face went up by at least three because every face in G is incident on one cousin-edge). Again, by Theorem 49, Definition 19, Theorem 48, and Lemma 52 we find that G' has a shortest-path-invariant addressing scheme with at most $5(q_{\max} - 1) - 6$ colors. Now color the non-cousin-edges of G according to their colors in G'. The cousin-edges can be colored with a single additional color since by Theorem 10, no vertex is incident on more than one cousin-edge. Therefore G can be colored with $5q_{\max} - 10$ colors. □

We conclude the Appendix with a couple of conjectures and a theorem related to addressing-schemes.

Conjecture 1. *Consider a regular tessellation $\{p, q\} \in \langle 3, 6\rangle \cup \langle 4, 4\rangle$. Then $\{p, q\}$ has a shortest path addressing-scheme with q colors for even p, and at most $q + 1$ colors otherwise.*

Theorem 53. *There exists an even-face-degree polyhedron that does not have a finite e-curve-coloring.*

Proof. Begin by drawing an infinite set of pairwise intersecting e-curves such there is one infinite-sided face bounded by an triangles and all other faces are squares. Now eliminate the infinite-sided face by adding and infinite set of e-curves that intersect every node of every triangle on the infinite-sided face boundary and continue to intersect the squares above the infinite face on the diagonals. Finally, draw an infinite set of parallel e-curves on the infinite face that run perpendicular to the other set of parallel e-curves just drawn. The induced tessellations is a half-plane square lattice adjoined to an infinite honeycomb lattice. The boundary between the two lattices is an alternating sequence of hexagons and squares. The polyhedron is an opposing edge polyhedron, but by our construction, the initial set of e-curves require an infinite set of colors. See Figure A.14 for the construction. □

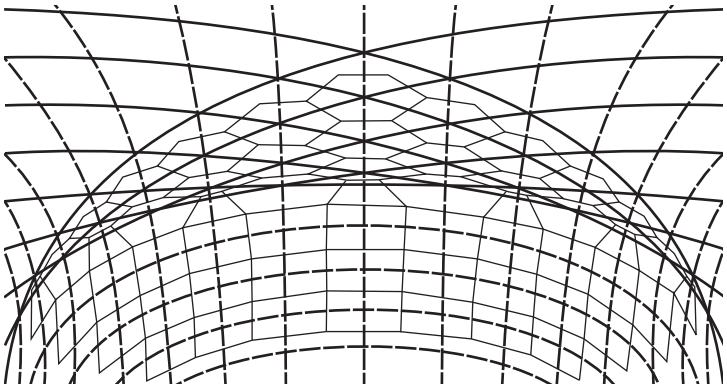

Figure A.14: A partially-drawn polyhedron and e-curves. The thinner lines represent the polyhedron and the thicker solid and dashed lines are the edgeset e-curves. The solid e-curve lines require an infinite number of colors to obtain an e-curve-coloring.

Conjecture 2. *There exists an even-face-degree polyhedron that does not have a finite shortest-path-invariant addressing-scheme and the polyhedron described in the preceding proof is an example of this.*

Bibliography

[1] James W. Anderson. *Hyperbolic Geometry*. Springer-Verlag, London, 1999.

[2] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays*, volume 2. Academic Press, 1982.

[3] Bernard Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, 2000.

[4] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, Dec 1996.

[5] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA, second edition, 2001.

[6] H. S. M. Coxeter and W. O. J. Moser. *Generators and Relations for Discrete Groups*. Springer-Verlag, Berlin, Heidelberg, Germany, fourth edition, 1980.

[7] Reinhard Diestel. *Graph Theory*. Springer-Verlag, New York, third edition, 2005.

[8] W. S. Evans. *Information Theory and Noisy Computation*. PhD thesis, University of California at Berkeley, 1994.

[9] W. S. Evans and L. J. Schulman. Signal propagation and noisy circuits. *IEEE Transactions on Information Theory*, 45(7):1–7, Nov. 1999.

[10] Tomás Feder. Reliable computation by networks in the presence of noise. *IEEE Transactions on Information Theory*, 35(3):569–571, 1989.

[11] Peter Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986.

[12] Peter Gács. Self-correcting two-dimensional arrays. In *Advances in Computing Research*, volume 5, pages 223–326. JAI Press, Greenwich, Conn., 1989.

[13] Peter Gács. Reliable cellular automata with self-organization. *Journal of Statistical Physics*, 103(1/2):45–267, 2001.

[14] Peter Gács and John Reif. A simple three-dimensional real-time reliable cellular array. *Journal of Computer and System Sciences*, 36(2):125–147, 1988.

[15] R. G. Gallager. *Information Theory and Reliable Communication*. Wiley, 1968.

[16] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, October 2006.

[17] S. Jimbo and A. Maruoka. Expanders obtained from affine transformations. In *STOC*, volume 17, pages 88–97, 1985.

[18] C. Kaklamanis, A. R. Karlin, F. T. Leighton, V. Milenkovic, P. Raghavan, S. Rao, C. Thomborson, and A. Tsantilas. Asymptotically tight bounds for computing with faulty arrays of processors. In *FOCS*, volume 31, pages 285–296, 1990.

[19] A. N. Kolmogorov. *Foundations of the Theory of Probability*. Chalsea Publishing Company, 1956.

[20] Joel L. Lebowitz, Christian Maes, and Eugene R. Speer. Statistical mechanics of probabilistic cellular automata. *Journal of Statistical Physics*, 59(1/2):117–170, April 1990.

[21] M. Margenstern. New tools for cellular automata in the hyperbolic plane. *Journal of Universal Computer Science*, 6(12):1226–1252, 2000.

[22] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 1995.

[23] Kihong Park. *Ergodicity and Mixing Rate of One-Dimensional Cellular Automata*. PhD thesis, Boston University, Boston, MA, USA, 1997.

[24] Nicholas Pippenger. On networks of noisy gates. In *26th Annual Symposium on Foundations of Computer Science (FOCS '85)*, pages 30–38, Los Angeles, Ca., USA, October 1985. IEEE Computer Society Press.

[25] Nicholas Pippenger. Reliable computation by formulas in the presence of noise. *IEEE Transactions on Information Theory*, 34(2):194–197, 1988.

[26] Nicholas Pippenger. Symmetry in self-correcting cellular automata. *Journal of Computer and System Sciences*, 49(1):83–95, August 1994.

[27] O. N. Stavskaya and I. I. Piatetski-Shapiro. On homogeneous nets of spontaneously active elements. *Systems Theory Research (transl. of Problemy Kibernetiki, Vol. 20, 1968)*, 20:75–88, 1976.

[28] A. L. Toom. Nonergodic multidimensional systems of automata. *Problems of Information Transmission*, 10(3):239–246, 1974.

[29] A. L. Toom. Monotonic binary cellular automata. *Problems of Information Transmission*, 12(1):33–37, 1976.

[30] A. L. Toom. Stable and attractive trajectories in multicomponent systems. In R. L. Dobrushin and Ya. G. Sinai, editors, *Multicomponent Random Systems*. Marcel Dekker, New York, 1980.

[31] S. Ulam. Random processes and transformations. In L. M. Graves, E. Hille, P. Smith, and O. Zariski, editors, *Proceedings of the International Congress of Mathematicians, 1950*, volume 2, pages 264–275, Providence, R. I., 1952. American Mathematical Society.

[32] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.

[33] J. von Neumann. *Theory of self-reproducing automata (compiled by A. W. Burks)*. Univ. of Illinois Press, Urbana, IL, 1966.