5-2002

Design and Simulation of Self-Organizing Microbial Cellular Automata

Michael Patrick Campfield
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Michael Patrick Campfield entitled "Design and Simulation of Self-Organizing Microbial Cellular Automata." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Bruce MacLennan, Major Professor

We have read this thesis and recommend its acceptance:

Stacy Prowell, Michael Berry

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:
I am submitting herewith a thesis written by Michael Patrick Campfield entitled “Design and Simulation of Self-Organizing Microbial Cellular Automata.” I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Bruce MacLennan
_____
Major Professor

We have read this thesis
and recommend its acceptance:

Stacy Prowell
_____

Michael Berry
_____

Accepted for the Council:

Dr. Anne Mayhew
_____
Vice Provost and
Dean of Graduate Studies

(Original signatures are on file in the Graduate Student Services Office.)

DESIGN AND SIMULATION OF SELF-ORGANIZING MICROBIAL CELLULAR AUTOMATA

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Michael Patrick Campfield

May 2002

Acknowledgments

There are several people who deserve acknowledgment in the creation of this thesis. Dr. Bruce MacLennan gave me the chance to show that computer science is still a living art.

Dr. Steven Ripp who provided, through his collection of journal articles, at least half of the research material needed and enough grounding to keep science from becoming science fiction.

Other people need to be acknowledged, without whom, I would not be attending this program. Dr. David Straight who allowed me into this program and my committee who took pity upon me.

Finally, I would like to acknowledge the progenitors, Christopher Theophanis, David Weston and Donald Thorp. Teachers who rarely get acknowledged but start so many of us on the greater journey toward understanding.

Abstract

This paper discusses the design and implementation of cellular automata based on the alteration of genetic sequences in bacteria. The work is composed of five chapters covering the problem, the system's design, the software simulation of the system and future issues on the problem. The section covering the problem explores the reasons for this work as well as issues that this work solves. The section covering system design details the modified genetic sequences and the algorithm that these sequences implement. The simulation section describes the layout of an experiment along with the test cases experimented on. Finally, the future work section points out lacking information from the work or possible difficulties this solution reveals.

Contents

List of Figures

List of Abbreviations

DNA	Deoxyribonucleic Acid
mRNA	Messenger RNA
NTPs	Ribonucleoside Triphosphate Molecules
rRNA	Ribosomal RNA
RNA	Ribonucleic Acid
tRNA	Transfer RNA

Chapter 1

Introduction

The current state of computer science and engineering is approaching an impasse. When observing the overall state of hardware and software development, the approach up to the current time was to create individual machines that were progressively more powerful, both in speed and processing capabilities. However, as the computing power of individual machines grows, so does the scope and depth of the problems offered up for analysis.

For example, in the 1970's most simulations of weather patterns and their future conditions were based on differential equations utilizing only a few variables on a localized region. As of the current year, algorithms used for simulating weather patterns commonly use differential equations with hundreds of variables based on a global scale. The problem of modeling weather patterns did not change, but the depth, in this case more variables, and the scope, the local region versus the world, did change.

This problem of increasing scope and depth in algorithm design has far exceeded the pro-

cessing ability of modern computer systems. While efforts have been made to interconnect groups of computers to accomplish large tasks in parallel, these systems are still limited by their inherent requirement that the problem be broken down into discrete problems that can be run in parallel. Some problems cannot be parallelized, or worse, parallelizing them results in algorithms whose complexity exceeds the complexity of the original problem.

While silicon based hardware may be limited in its ability to handle problems of excessive scope and depth, the solution may lie in organic systems. Organic systems, such as cells, bacteria, and DNA, have demonstrated that they are well suited for the task of computation. From the organization and operation of neurons in a brain to the individualization and specialization of stem cells in a growing embryo, organic components show a great promise for exploitation in solving computing problems.

1.1 Cellular Automata

The first cellular automata were developed by von Neumann and Ulam in the 1940's and 1950's in order to study the behavior of complex systems [15, 2, 13]. Since then they have developed into their own specialized field of study. In essence, cellular automata can be described as spatially dependent machines. This is due to the fact that their operation is an algorithm over an N-dimensional lattice. Cellular automata have several interesting characteristics beyond the ability to solve any Turing-solvable problem; notable among them is their relation to classic differential equations [8].

The basic unit of a cellular automaton is the "cell" or "site" which is simply a holder of one

or more independent state values. Each cell is in contact with a fixed number of other identical cells which define its neighborhood. Operating over the entire system is a set of transition rules. These transition rules define how the state of each cell will change during a cycle of the automaton based on the current state of the cell and the states of its neighbors.

In one version of a classic example, Conway's "Game of Life," [1] the state of the cell is based on the cell states of its eight neighbors. If four or more neighbors' states are *ON* or if there are less than two neighbors who are *ON* then the current cell's state becomes *OFF*. If a cell has three neighbors that are *ON* then its own state will become *ON*. This is one of the simplest state transition tables available, because each cell in an automaton can have more than one independent state value resulting in an significant number of possible transition mappings. However, for simplicity the majority of cellular automata rely on a single state value per cell.

1.2 Recent History of Biomolecular Computation

There have been three recent events which have re-introduced organic computation as a viable method for algorithm solution. The first of these events was the publication of the paper "Engineered Communications for Microbial Robotics" by Weiss and Knight [16]. Weiss and Knight's paper summarized the work to date and provided new directions for study based upon recent work in biology. The second event was the development of new methods for manipulating DNA sequences with reliably reproducible results. These new methods for DNA strand manipulation gave rise to the engineering of DNA sequences that generated desired protein combinations with such factors as polymerase hindrance and promoter manipulation [10]. The third event,

the systematic identification of a large group of pre-existing DNA encodings, introduced a set of building blocks that could be manipulated to form protein regulators used to control the operation of biological computation devices [17].

1.3 Primary Approaches to Biomolecular Computation

1.3.1 DNA Based Systems

While most biomolecular systems rely on manipulated sequences of DNA to control cell activity, these computational systems operate directly on the physical and chemical interaction of DNA molecules. The benefits of these systems are that they provide significant data density and massively parallel operation.

Lattice Systems

DNA lattice systems use a vessel containing engineered DNA strands to create a three-dimensional lattice-based structure within the container. This self-assembling lattice is generated by the formation of stable geometric patterns interlinked by strand exchange. This lattice can be used to execute dynamic algorithms and produce solutions for any Turing-solvable problem [5, 12].

Search Systems

DNA search systems rely on the massively parallel operation provided by the simultaneous interaction of billions of DNA strands. This method of DNA searching can be used to search for solutions to any NP problem in polynomial time based upon the operations outlined by

Adleman and Lipton [5, 6, 7, 11].

1.3.2 Single Cell Systems

Single cell systems employ individual microbes to execute localized algorithms. Specifically, most of these systems implement basic boolean functionality using gene expression. These systems are considered localized because they rely on no external influences, excluding the input vector represented by the presence or lack of signal chemicals [3]. Single cell systems are not considered practical because of their slow speed, limited ability to handle complex algorithms and the inability to recover from system failure, i.e. cell death. However, these systems are commonly employed as the base elements in multi-cellular systems.

1.3.3 Multi-Cellular Systems

Multi-cellular systems are the most promising of the three biomolecular systems in terms of practicality utilizing current technology. These systems employ the direct interaction of two or more cells to accomplish computation. While multi-cellular systems can be of one, two or three dimensions, this paper concerns itself with multi-cellular systems of only two dimensions. Multi-cellular systems may be visualized as an eight-neighbor tiling. These systems employ cells that act as individual processing elements with the ability to communicate with their neighbors. This communication, both within the local neighborhood and over the entire cellular field, permits the individual processing elements to interact, increasing the speed and reliability of the computation while greatly expanding the range of executable algorithms. Furthermore, multi-cellular systems lend themselves to being employed as cellular automata. Their

design closely mimics the proto-typical design that CA's take, a "cell" system with interactions between neighboring elements.

1.4 Current Issues in Multi-Cellular Systems

As with all biomolecular systems, current multi-cellular systems have notable limitations. In order to create a practical microbial based cellular automata, a cellular tiling of a reasonable size must be created. The difficulty arises from the natural inability for unspecialized cells, such as bacteria, to discriminate signals from neighboring cells. Without this feature, it is impossible to implement a state transition table that exceeds two transitions, based on any neighbor on or all neighbors off. In order to fully implement useful biomolecular automata there must be a method to self-organize the tiling into distinct structures that rely on these two state transitions or enable cellular structures to identify distinct chemical signals emanating from local neighbor cells. This paper explores a system based on modified bacteria that after three stages of initialization, results in a four-way independent cellular structure capable of recognizing distinct chemical signals from a four-neighborhood. The self-tiling nature of the proposed system allows for a reasonable size automaton to be created with minimal interaction from an outside source.

Chapter 2

Automata Design

2.1 Overview of Biological Concepts

2.1.1 Bacterial Films

This cellular automaton operates over a synthetic bacterial film. Several challenges arise when utilizing a living organism for computation, the first of which is the irregular packing of the bacteria. While it is simple to produce an engineered bacterial film on agar, it is difficult to force the bacteria into a regular pattern.

While some work has been done by Mike Simpson, a researcher from Oak Ridge National Laboratories, in producing carbon nanofibers which act as binding locations for bacteria, this work is still far from practically exploitable [9].

The solution to the regularity problem is to create a bacterial film on a medium that reduces movement, but resolve the regularity issues in the tiling process and algorithm design.

2.1.2 Protein Synthesis

Proteins are generated by gene expression, a two part process based on the transcription and translation of RNA. To accomplish RNA transcription, the primary operational component of the process, RNA polymerase must identify and connect to the correct starting point along a DNA strand. This point is a sequence of approximately 40 base-pair units of DNA called the promoter. Once the polymerase has identified the transcription starting point, it binds to the DNA strand and begins to unwind the DNA within its local area. As it unwinds the molecule it uses available NTPs to create a RNA strand from the template section of the unwound DNA. This RNA strand continues to form as the polymerase travels down the DNA molecule, unwinding, transcribing to the RNA strand and then rewinding the DNA. The process terminates when the polymerase reaches a terminator sequence in the DNA. This termination sequence causes the polymerase to finish RNA transcription and disconnect from the DNA molecule. The RNA molecule can incur subsequent chemical modification following separation from the polymerase. Three forms of RNA can be generated, rRNA, tRNA and mRNA. While the method of their creation after transcription is significant, it is outside the range of this study.

The process of translation begins with the separation of the completed RNA strand from the polymerase and its subsequent conversion to mRNA. The mRNA strand interacts with three primary factors, two ribosomal subunits and numerous tRNA molecules carrying amino acids, the building blocks for the protein. The two ribosomal subunits combine with the mRNA strand along a starting sequence on the mRNA. During the translation phase, the ribosomal subunit travels down the mRNA strand. The subunits select amino acid bearing tRNA molecules based

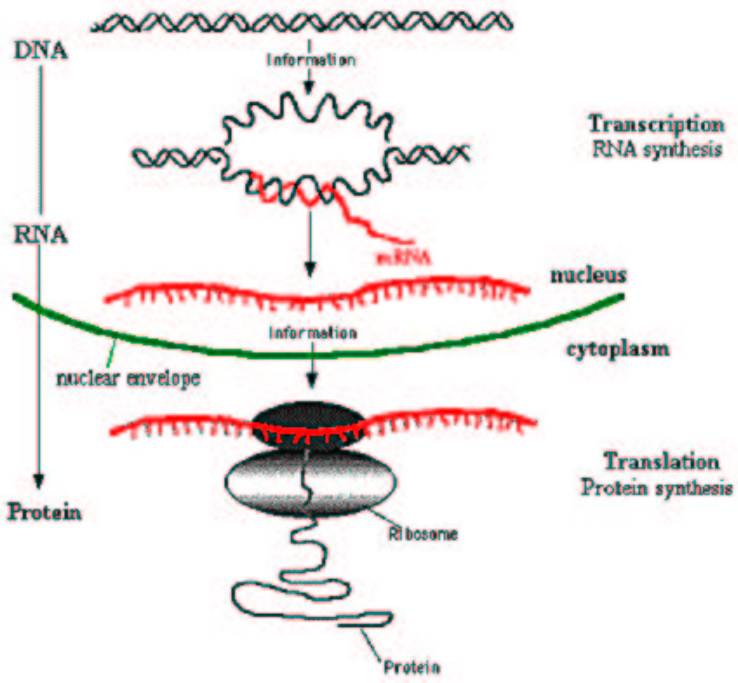

Figure 2.1: Protein Synthesis Cycle [14]

upon the current mRNA location. These tRNA molecules divest their amino acids into a growing polypeptide chain, the protein. Upon reaching a release point on the mRNA, the ribosome structure releases the protein and mRNA molecule. The entire process is illustrated in figure 2.1.

One notable feature of gene expression is the generation of multiple proteins through one transcription and translation cycle. This process occurs naturally when multiple proteins must be produced simultaneously to preserve chemical balance within the cells.

2.1.3 Controlling Factors

The process of gene expression that results in protein generation has several regulating factors. The primary control methods utilized in this cellular automata model and in other biomolecular systems is polymerase repression and promotion. Polymerase repression involves introducing a protein or set of proteins that bind to the DNA molecule at a point located between the start and end of the gene responsible for the generation of a mRNA strand called the operator site. When a protein is bound to the operator site of the DNA, the polymerase can not physically traverse the full length of the gene, thus halting the transcription of mRNA and subsequent translation into a protein. Polymerase promotion relies on utilizing promoter sites with limited ability to attract and bind polymerase. These weak promoters require additional activator proteins that bind to the DNA molecule and make the promoter site active. Without these proteins the gene will lie dormant, unable to bind polymerase. By selective application of the activator proteins, genes can be induced into activation. Numerous proteins able to halt or promote polymerase transcription have been identified and by using genetic manipulation can be selectively inserted within the genetic strand. Once of the simplest examples of this genetic regulation is the *lac* operon of E.coli as seen in figure 2.2 [10].

2.1.4 Intercellular/Intracellular Signals

The use of proteins to regulate cellular processes is primarily confined within the boundaries of the cell wall. However, for multi-cellular cellular automata there must exist two scopes for chemical signaling, an intracellular and intercellular set of proteins. This differentiation of pro-

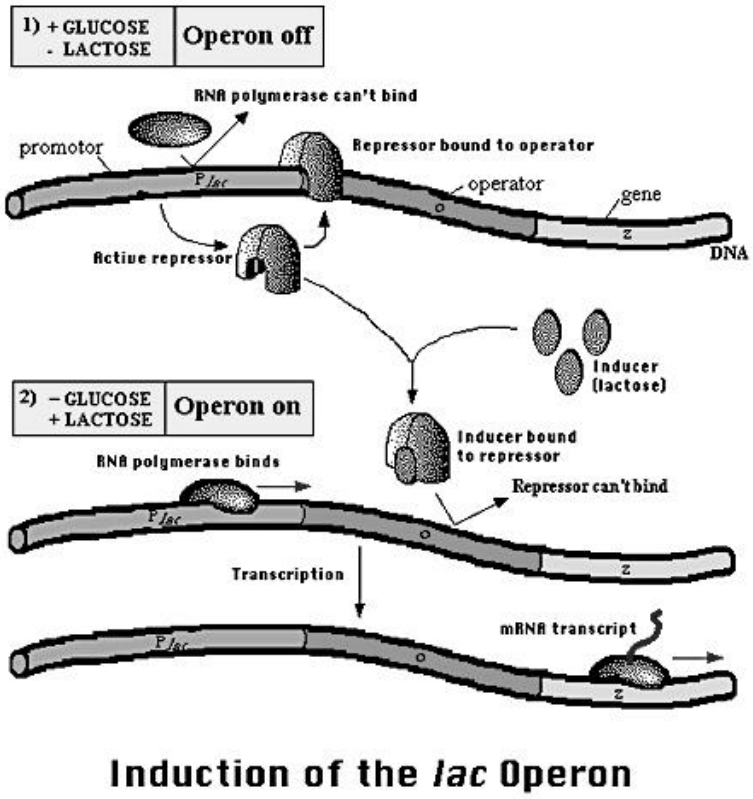

Figure 2.2: *lac* Operon Regulation [4]

tein classes allows localized control operations within the cell walls while having a secondary set of proteins that are able to traverse cell walls thereby allowing for signaling within the local cell neighborhood. This is not completely feasible because intracellular proteins are generated within the cell body and threfore will be able to interact with local DNA as intercellular proteins do. However this difficulty can be compensated for within the local cellular environment.

2.2 Automata Organization

The operation of the bacterial automaton is based on two phases, a structural setup phase and the phase during which computation occurs. The first phase involves the establishment of four-neighbor signal independence in a bacterial matrix that has the regular tiling as seen in figure 2.5. on page 24. The described bacterial matrix is an idealized model of the system. In practice the bacterial film will be an irregular, densely packed layer. Without modification to the film, any system requiring a perfectly regular tiling would fail. However, it is possible to introduce perceived regularity by forcing certain units to be chemically inactive, and therefore ignored. Using this perceived regularity, the structural setup proceeds in three stages: one-way, two-way and four-way independence, described in sections 2.3, 2.4, and 2.5, respectively.

The second phase is the operation of the biological film as a multi-element cellular automaton. This involves initializing the automaton's border cell values, then allowing the computation to progress and recording values following a set duration or when a relative equilibrium is reached.

2.2.1 Selection of Signaling Proteins

The available selection of proteins usable for control and signaling within the bacterial tiling is significant. However, the proteins selected must have a well defined scope of operation: intercellular verses intracellular. Furthermore, these proteins cannot be proteins naturally present in the cell's environment. This will prevent erroneous signal crosstalk between normal cell regulatory process and the artificial cellular automata signals.

The rate of protein synthesis through transcription and translation is not identical for all proteins. Some proteins, due to complexity or composition have a higher rate of synthesis compared to other proteins. The synthesis rate for proteins that are operating in conjunction or opposition to other proteins must be similar in order to prevent undesired state changes.

Finally, the proteins must all have the ability to promote or repress polymerase transcription with known repression sites. With these limits, it is possible for hundreds of different proteins to be employed as control and signaling systems. For this reason, this thesis refers to proteins and repressor binding sites as abstract values to be decided by the genetic engineer at the time of the DNA alteration.

2.2.2 Notation

The notation system for the cellular automaton is designed to remove unnecessary complexity from the system, focusing on the interactions of proteins with the DNA in regulating the system. Proteins are designated by a Greek, English or Hebrew symbol and scope. For example, β_I denotes a protein, β, that is local to the cellular environment. α_X denotes a protein, α, that

exists in the cell and can move outside of the cell body. The notation for gene sequences follows in a similar fashion:

$\textit{Promoter}_{Activator\,Protein} \mid \textit{Operator}_{Repressor\,Protein} \ldots \mid$ *Proteins Generated*

Any promoter sequence, designated *P*, with the $\emptyset$ subscript indicates the promoter does not require an activating protein to function. The $\emptyset$ also denotes that there are no corresponding operators in the operator section of the gene. All notation is considered case sensitive. For example, the protein Ω is not the same as ω.

2.2.3 Caveats

The first caveat is that the physical range of the protein selected will be four-neighbor specific. Transmission of proteins across the film is probabilistic in nature. By the proper selection of proteins we can guarantee with a certain reliability that successful transmission of proteins will occur to only the four neighboring cells before the proteins are utilized or degraded.

The second caveat is that the tiling rate of selected sections of the film will not be significantly different from any other section. This rate, controlled by bacterial activity and the intercellular protein production rates prevents rapid tiling in one section that could lead to extended error correcting sequences.

2.3 One-Way Independent Tiling

One-way independent tiling takes the uninitialized film and produces an alternating tiling of bacteria with two different protein production states, α positive or β positive, where being

chemically positive indicates that their is production of said chemical occuring and the concentration of the chemical is measurable. The two gene sequences responsible for regulating the tiling and producing the intercellular communications system are:

1. $P_\gamma \mid \alpha\, \epsilon\, \Omega \mid \beta_I \beta_I\, \delta_X$

2. $P_\delta \mid \beta\, \zeta\, \Omega \mid \alpha_I\, \alpha_I \gamma_X$

These and some later gene sequences have multiple productions of certain proteins. The reasoning for this is that these proteins are used so extensively in many sequences that an adequate concentration must be maintained.

2.3.1 Initialization Sequence

The tiling sequence is initialized when the protein γ or δ is introduced into the film at the point $(0, N)$ as designated in figure 2.3. For this example, let the introduced protein be γ. Prior to this event, the cell has no state since production of α and β require activation proteins. However, upon its introduction, γ binds to the promoter in gene sequence 1 and acts as the activation protein for the polymerase. The transcription and translation phases, proceed with no delay generating β_I and δ_X. β binds to the operator of sequence 2 and δ binds to the promoter sequence. The polymerase can now attach to sequence 2, but the bound β protein prevents transcription of the sequence. The primary local protein is now β with no α's in the environment. This cell is now considered β positive. Certain percentages of the protein δ exit the local cell environment and proceed to the bounding neighbors of the cell, the cells $(0, N-1)$ and $(1, N)$. The δ proteins initialize the environment of these cells as the last cell was initialized

by γ. These two cells become α positive and continue the progression of tiling across the film.

2.3.2 Error Correction

Four gene sequences provide for error correction in the one-way tiling sequence:

3. $P_{\emptyset} \mid \gamma \mid J_I$

4. $P_{\delta} \mid \alpha\ \beta\ J \mid \Omega_I$

5. $P_{\eta} \mid \zeta\ \beta\ \Omega \mid \epsilon_I \epsilon_I\ \theta_X$

6. $P_{\theta} \mid \epsilon\ \alpha\ \Omega \mid \zeta_I\ \zeta_I \eta_X$

The first two sequences, 3 and 4, are critical to creating the perceived regularity of the bacterial film. The rule expressed is that if the cell has no state and it is receiving both one-way intercellular signals, γ and δ, the cell will generate Ω. Ω denotes a global halting protein. This protein has repressor sites on key genetic sequences that if blocked, will effectively deactivate the bacteria's automata nature. By causing cells that are within the zone of distribution of both intercellular proteins to become inactive, a pattern will emerge in the film where each cell is bounded by only active neighbors of the opposite state in the one-way tiling.

The second two sequences, 5 and 6, function identically as sequences 1 and 2. However, their initialization begins with the introduction of the activator protein at $(0, 0)$ from figure 2.3. The value of the film's axes lengths, N, can be any integer value greater than zero, but the initial activator protein must be selected based on the state of the cell located at $(0, 0)$ following the initialization of the first two sequences. This provides error correction for the system since

(0,0)

β	α	β	α	β	α	β	α	β
α	β	α	β	α	β	α	β	α
β	α	β	α	β	α	β	α	β
α	β	α	β	α	β	α	β	α
β	α	β	α	β	α	β	α	β
α	β	α	β	α	β	α	β	α
β	α	β	α	β	α	β	α	β
α	β	α	β	α	β	α	β	α
β	α	β	α	β	α	β	α	β

(0,N)

Figure 2.3: One-Way Tiling

the two one-way tilings share blocking proteins and each will only initialize if the other is correct. Otherwise, the cells that conflict will enter a non-production state until a valid protein combination is introduced.

While sequences pairs 1, 2 and 5, 6 produce similar results and can be considered redundant, the transcription blockers shared by the sequences cause the systems to interact in a mutually exclusive operation. An initialized cell will be of the state $\alpha\epsilon$ positive or $\beta\zeta$ positive. If the cell is in any other state the mutual transcription blocking mechanism will halt all state productions until a viable state is introduced. In terms of error correction, if any section of the film is disrupted by die-off, replication or physical shock, and this disruption alters the tiling into an incorrect state pattern, the cells on the boundary of this disruption will revert into a condition

of no state. As time progresses, the bacteria re-initialize as valid states, taking input from their neighbors. Difficulty arises if the physical shock alters a significant section of the bacterial film, causing it to spend indefinite time in an invalid or unset state, as the the sections of film conflict.

2.4 Two-Way Independent Tiling

Two-way independent tiling is a further refinement of an already established one-way independent tiling. While one-way independent tiling produces an alternating pattern in the film, the usefulness of this in intercellular communication is limited. The cell at the center of the neighborhood can only receive messages from his four neighbors without telling the location of the signal producer. Two-way independence allows a cell to discern two signal patterns from its four neighbors. Two cells share one protein state and the other two share a different protein state. With modification, this form of tiling can be used to produce automata with four transition states: no neighbors active, one pair of equal state neighbors active, or both pairs of equal state neighbors active. The six genetic sequences are:

7. $P_{\pi} \mid \beta\ \Omega\ \lambda\ \sigma\ o\ \rho\ \psi \mid \iota_I \iota_I\ \nu_X$

8. $P_{\nu} \mid \alpha\ \Omega\ \mu\ \phi\ \rho\ \psi\ \pi \mid \kappa_I \kappa_I\ \xi_X$

9. $P_{\xi} \mid \beta\ \Omega\ \sigma\ \iota\ \nu\ \psi\ \pi \mid \lambda_I\ \lambda_I\ o_X$

10. $P_{o} \mid \alpha\ \Omega\ \kappa\ \phi\ \pi\ \xi\ \nu \mid \mu_I\ \mu_I\ \rho_X$

11. $P_{\rho} \mid \beta\ \Omega\ \iota\ \lambda\ o\ \xi\ \nu \mid \sigma_I\ \sigma_I\ \psi_X$

12. $P_\psi \mid \alpha\ \Omega\ \kappa\ \mu\ o\ \xi\ \rho \mid \phi_I\ \phi_I\ \pi_X$

These sequences are inserted into the genetic code with the first six sequences for one-way tiling. While a two-way independent tiling is possible with only three distinct states, without the final three states the film can initialize incorrectly due to small tiling rate differences between cells. This error can be caused if a set of cells initializes around an uninitialized group of boundry cells causing them to change state. This state change may be incorrect since the previously uninitialized cells do not yet possess the blockers required to prevent unwanted intercellular proteins from influencing the state. This will be referred to as a loop-back error. The addition of the final three sequences also leads to the simplification of the four-way tiling as shall be seen later.

2.4.1 Initialization Sequence

When the one-way tiling has reached a point of minimal error, the two-way tiling is initialized producing a bacterial tiling with one of six different protein production states, ι, κ, λ, μ, σ, or ϕ positive. The initialization sequence starts with the completion of the initialization of the one-way tiling. When the primary film tiling is completed the second of the initialization proteins is inserted at $(0, N)$. Since this cell is β positive through the original one-way tiling sequence, the initialization protein must be ν, o or ψ, in this example it is ν. Once this chemical is introduced into the film, the two-way tiling progresses as with the one-way tiling forming the pattern in figure 2.4.

(0,0)

μ	σ	ϕ	ι	κ	λ	μ	σ	ϕ
λ	μ	σ	ϕ	ι	κ	λ	μ	σ
κ	λ	μ	σ	ϕ	ι	κ	λ	μ
ι	κ	λ	μ	σ	ϕ	ι	κ	λ
ϕ	ι	κ	λ	μ	σ	ϕ	ι	κ
σ	ϕ	ι	κ	λ	μ	σ	ϕ	ι
μ	σ	ϕ	ι	κ	λ	μ	σ	ϕ
λ	μ	σ	ϕ	ι	κ	λ	μ	σ
κ	λ	μ	σ	ϕ	ι	κ	λ	μ

(0,N)

Figure 2.4: Two-Way Tiling

2.4.2 Error Correction

Each of the six gene sequences composing the two-way tiling have several built in error correction systems. The primary error correction system is the α and β blockers. While the two-way tiling is theoretically independent of the one-way tiling, the α and β blockers prevent local neighborhood errors by patterning the two-way tiling on the correct tiling of the one-way tiling. For example, the only valid states for a cell which is α positive are ι, σ and λ. Any attempt to produce κ, μ, or ϕ will be prevented by the binding of α on their respective genetic sequences.

The secondary error correction system comes through the use of intercellular proteins as protease inhibitors These sequences are part of the primary control mechanism for the tiling itself, but once the tiling is completed the sequences force the bacterial film to remain in the same state pattern in a method similar to the one-way independent tiling. If any initialized cell is not bounded by the two specific cell states permissible by that state, its state is reset until it finds a new valid state, or the states of its neighbors change.

2.5 Four-Way Independent Tiling

Four-way independent tiling is the final refinement of the film in the computational automaton. Following the completion of two-way tiling, the four-way tiling is initialized to produce the set of nine protein production states, ח,ז,ו,ה,ד,ג,ב,ק, or ט positive. By utilizing nine protein units, we guarantee that each cell of a certain state will have no equal within an eight-neighborhood, reducing the probability of error. Furthermore, as with the two-way tiling, the system does not require the full number of genetic sequences to introduce tiling. However, as in the two-

way tiling, by increasing the number of sequences, the possibility of tiling errors due to small tiling rate differences is nearly eliminated. By guaranteeing that each cell has no equal within an eight-neighborhood, each cell will be surrounded by a four-neighborhood with each cell existing in a unique protein state.

In order to generate these unique states, the tiling requires a layered pattern as shown in figure 2.5. During the two-way tiling, if only ι, κ and λ were utilized, the tiling would not be diagonally regular, a property we exploit to reduce the number of required states in generating four-way independence.

The eighteen modified gene sequences required are:

13. $P_\Gamma \mid \beta\ \iota\ \sigma\ \Lambda\ \Pi\ \Xi \mid$ קָ $_I$ קָ $_I\ \Delta_X$

14. $P_\Gamma \mid \beta\ \lambda\ \sigma\ \Lambda\ \Pi\ \Xi \mid$ בָּ $_I$ בָּ $_I\ \Delta_X$

15. $P_\Gamma \mid \beta\ \iota\ \lambda\ \Lambda\ \Pi\ \Xi \mid$ גָּ $_I$ גָּ $_I\ \Delta_X$

16. $P_\Delta \mid \alpha\ \kappa\ \mu\ \Gamma\ \Pi\ \Xi \mid$ דָ $_I$ דָ $_I\ \Theta_X$

17. $P_\Delta \mid \alpha\ \kappa\ \phi\ \Gamma\ \Pi\ \Xi \mid$ הָ $_I$ הָ $_I\ \Theta_X$

18. $P_\Delta \mid \alpha\ \mu\ \phi\ \Gamma\ \Pi\ \Xi \mid$ וָ $_I$ וָ $_I\ \Theta_X$

19. $P_\Theta \mid \beta\ \iota\ \sigma\ \Gamma\ \Pi\ \Delta \mid$ זָ $_I$ זָ $_I\ \Lambda_X$

20. $P_\Theta \mid \beta\ \lambda\ \sigma\ \Gamma\ \Pi\ \Delta \mid$ חָ $_I$ חָ $_I\ \Lambda_X$

21. $P_\Theta \mid \beta\ \iota\ \lambda\ \Gamma\ \Pi\ \Delta \mid$ טָ $_I$ טָ $_I\ \Lambda_X$

22. $P_{\Lambda} \mid \alpha\ \kappa\ \mu\ \Gamma\ \Theta\ \Delta \mid$ ק$_I$ ק$_I$ Ξ_X

23. $P_{\Lambda} \mid \alpha\ \kappa\ \phi\ \Gamma\ \Theta\ \Delta \mid$ ב$_I$ ב$_I$ Ξ_X

24. $P_{\Lambda} \mid \alpha\ \phi\ \mu\ \Gamma\ \Theta\ \Delta \mid$ ג$_I$ ג$_I$ Ξ_X

25. $P_{\Xi} \mid \beta\ \iota\ \lambda\ \Lambda\ \Theta\ \Delta \mid$ ו$_I$ ו$_I$ Π_X

26. $P_{\Xi} \mid \beta\ \iota\ \sigma\ \Lambda\ \Theta\ \Delta \mid$ ד$_I$ ד$_I$ Π_X

27. $P_{\Xi} \mid \beta\ \sigma\ \lambda\ \Lambda\ \Theta\ \Delta \mid$ ה$_I$ ה$_I$ Π_X

28. $P_{\Pi} \mid \alpha\ \phi\ \mu\ \Xi\ \Theta\ \Lambda \mid$ ט$_I$ ט$_I$ Γ_X

29. $P_{\Pi} \mid \alpha\ \kappa\ \mu\ \Xi\ \Theta\ \Lambda \mid$ ז$_I$ ז$_I$ Γ_X

30. $P_{\Pi} \mid \alpha\ \phi\ \kappa\ \Xi\ \Theta\ \Lambda \mid$ ח$_I$ ח$_I$ Γ_X

2.5.1 Initialization Sequence

As with one-way and two-way independent tiling, four-way tiling initializes at the end of the previous tiling sequence. Four-way tiling is begun with the introduction of an extracellular protein, Γ, Δ, Θ or Λ at position (N, N) from figure 2.4. The selection of the protein is based on the initialized state of the cell at (N, N) which has been fixed during the two-way tiling. For this example, the protein selected is Γ. The progression of the four-way tiling is remarkably similar to the progression of two-way tiling with triple the number of genetic encodings to compensate for the presence of eighteen operating states as compared to the six states used in

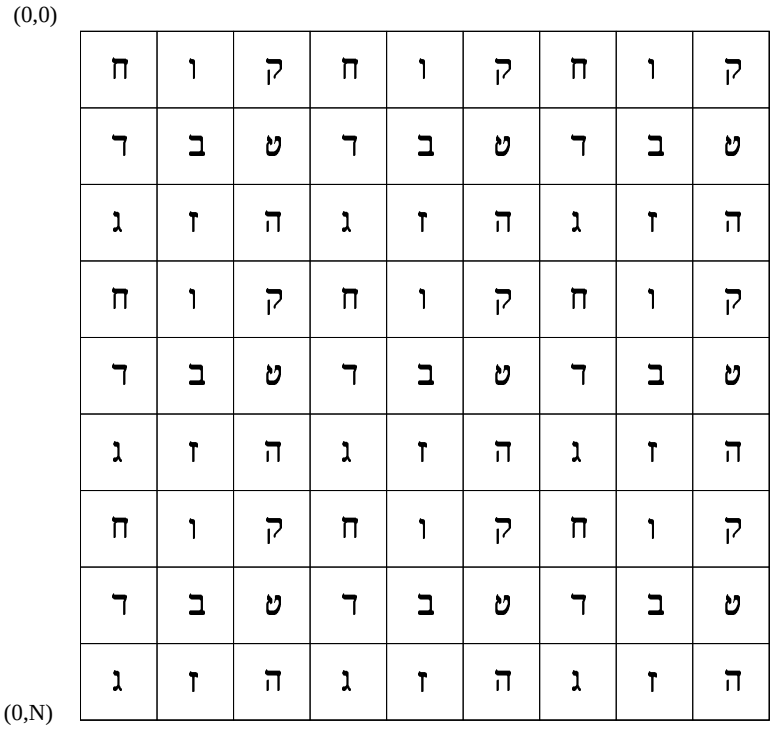

Figure 2.5: Four-Way Tiling

the two-way tiling. At the completion of the tiling sequence the film will have a state mapping similar to the one seen in figure 2.5.

2.5.2 Error Correction

Error correction in the four-way tiling is based on three elements. The primary error correction system is the polymerase inhibiting sequences in the eight encodings. As in the one-way and two-way tilings these sequences initialize the tilings but also maintain the tiling's order. The secondary error correction system, as in the two-way tiling system, is the utilization of the intercellular proteins to limit valid states that may co-exist in cells.

2.5.3 Summary

Each of the three tiling phases requires time, proportional to the area of the film, to complete. This time period is further modified by the potential for errors to encroach into the tiling, requiring the error correction schemes time in resetting the local area.

Upon completion of the four-way tiling process, the film has several unique properties. The primary property introduced is the nature of four-way independence. Each cell within the tiling has a state that is unique from the states in its eight-neighborhood. Therefore, each cell can independently identify intercellular signals based on the independent states of it and its neighbors.

2.6 Automata Computation

Upon completion of the structural setup phase, the automaton is prepared for the computational phase. For computation, the automaton must be able to exploit the four-way independent film tiling to identify the individual cell signals relative to their state. The method the automaton uses is a set of protein control genes similar to the controls used in the initial tiling phase. The control sequence for cellular processes is separated into four phases: running state identification, computational signal generation, signal translation, and transition mapping.

2.6.1 Notation

The notation for the control sequences has two modifications on the original notation system. The primary variation is that individual proteins are identified with Roman characters instead of

Greek characters. The second modification is that the proteins used to designated *ON* and *OFF* are denoted א and ∇ respectively.

2.6.2 Running State Identification

Running state identification refers to the state of the cell as it relates to the operation of the cellular automaton along with the input and output of values from the system. This state, *ON* or *OFF* denoted by the presence of the proteins א or ∇, is the true controlling factor of the operation of the automata. In theory, the operating film will have a set of input paths able to introduce these proteins into the film or to detect the presence of this chemical at points within the system. There are two control genes for this phase.

31. $P_M \mid Y \mid \text{א}_X \text{א}_X$

32. $P_\emptyset \mid \text{א} \mid \nabla_X \nabla_X \; Z_I Z_I Z_I$

The first of these sequences is the *ON* control sequence. Its operation is simple, the reception of the protein M, described later, produces the protein א. The second sequence is the *OFF* control sequence. By default the cell is considered *OFF*. The presence of א halts production of ∇ and Z. ∇ is the *OFF* signal protein while Z, like Ω, is a halting signal present in several later sequences.

2.6.3 Computational Signal Generation

Signal generation is the set of gene sequences that control the production of proteins based on the state of the cell, א or ∇, and the state designated by the tiling, ח,ז,ו,ה,ד,ג,ב,ק or ט.

There are eight sequences that control computational signal generation:

33. $P_{\text{ק}} \mid Z \mid A_X$

34. $P_{\text{ב}} \mid Z \mid B_X$

35. $P_{\text{ג}} \mid Z \mid C_X$

36. $P_{\text{ד}} \mid Z \mid D_X$

37. $P_{\text{ה}} \mid Z \mid E_X$

38. $P_{\text{ו}} \mid Z \mid F_X$

39. $P_{\text{ז}} \mid Z \mid G_X$

40. $P_{\text{ח}} \mid Z \mid H_X$

41. $P_{\text{ט}} \mid Z \mid K_X$

The cell's tiling based state initiates transcription on the appropriate gene sequence. If the cell's overall state is *ON* then the transcription and translation proceeds normally generating the intercellular proteins relative to that cell. However, if the cell's state is *OFF*, as is by default, the proteins transcription is inhibited resulting in no signal protein generation. Through this, the genes operate as individual signalers to neighboring cells. If a cell's state is ק and the cell is in an *ON* state, then the cell will produce A. If the cell is in the *OFF* state, then the transcription will be halted and A will not be present. Because each neighbor in the tiling is of a different state, ח,ז,ו,ה,ד,ג,ב,ק, or ט, each cell's neighbors will produce a different *ON* signal chemical which can be counted.

2.6.4 Signal Translation

Signal translation is the set of gene sequences that detect the presence of computational signals and use them as blocking proteins for gene sequences producing intracellular proteins. This step reduces the complexity of the transition mapping, diminishing the number of gene sequences required for a set of rules.

42. $P_{\emptyset} \mid A \mid N_I$

43. $P_{\emptyset} \mid B \mid O_I$

44. $P_{\emptyset} \mid C \mid Q_I$

45. $P_{\emptyset} \mid D \mid R_I$

46. $P_{\emptyset} \mid E \mid S_I$

47. $P_{\emptyset} \mid F \mid T_I$

48. $P_{\emptyset} \mid G \mid U_I$

49. $P_{\emptyset} \mid H \mid V_I$

50. $P_{\emptyset} \mid K \mid W_I$

The sequences' operation is simple, halting the production of intracellular signals if an intercellular signal is present.

2.6.5 Transition Mapping

The transition mapping defines how the cellular automaton will operate based on the rules encoded into the gene sequences. It is possible to encode the transition table in a myriad of ways, creating hundreds of local control structures based on repression or promotion of different neighbor signals. However, this example automata's transition mapping will simulate a variation of Conway's "Game of Life" as a good example of cellular independence. The "Game of Life" has two rules:

1. If a cell is surrounded by 2 or 3 *ON* neighbors the cell will become *ON*

2. If a cell is surrounded by only 1 or 4 *ON* neighbors the cell will become *OFF*

These rules are expressed through a set of twenty-seven protein sequences that exploit the products of the signal translation. By default, the proteins generated by signal translation are always present, halting production of M and keeping the cell state *OFF*. However, if two or more computational signals are present that are from the same neighborhood grouping, the signal translation proteins are stopped and one of the eight gene sequences progresses to M generation resulting in the change of running state identification to *ON*.

The first eighteen genetic sequences are designed to detect if two or three neighbors are active. Each sequence has blockers for locally produced signal proteins in order to prevent counting the local cell's state. The final nine sequences test for the presence of four active neighbors, producing the protein Y, which blocks the production of א.

51. $P_{\emptyset} \mid N\ O\ \text{ך}\ \text{ב} \mid M_I$

52. $P_\emptyset$ | $N\ R$ ך ד | M_I

53. $P_\emptyset$ | $N\ T$ ך ו | M_I

54. $P_\emptyset$ | $N\ V$ ך ח | M_I

55. $P_\emptyset$ | $O\ Q$ ב ג | M_I

56. $P_\emptyset$ | $O\ T$ ב ו | M_I

57. $P_\emptyset$ | $O\ W$ ב ט | M_I

58. $P_\emptyset$ | $Q\ R$ ג ד | M_I

59. $P_\emptyset$ | $Q\ S$ ג ה | M_I

60. $P_\emptyset$ | $Q\ W$ ג ט | M_I

61. $P_\emptyset$ | $R\ S$ ד ה | M_I

62. $P_\emptyset$ | $R\ V$ ד ח | M_I

63. $P_\emptyset$ | $S\ T$ ה ו | M_I

64. $P_\emptyset$ | $S\ U$ ה ז | M_I

65. $P_\emptyset$ | $T\ U$ ו ז | M_I

66. $P_\emptyset$ | $U\ V$ ז ח | M_I

67. $P_\emptyset$ | $U\ W$ ז ט | M_I

68. $P_\emptyset$ | $V\ W$ ח ט | M_I

69. $P_{\emptyset} \mid O\ R\ T\ V \mid Y_I$

70. $P_{\emptyset} \mid N\ Q\ T\ W \mid Y_I$

71. $P_{\emptyset} \mid O\ R\ S\ W \mid Y_I$

72. $P_{\emptyset} \mid N\ Q\ S\ V \mid Y_I$

73. $P_{\emptyset} \mid Q\ R\ T\ U \mid Y_I$

74. $P_{\emptyset} \mid N\ O\ S\ U \mid Y_I$

75. $P_{\emptyset} \mid S\ T\ V\ W \mid Y_I$

76. $P_{\emptyset} \mid N\ R\ U\ W \mid Y_I$

77. $P_{\emptyset} \mid O\ Q\ U\ V \mid Y_I$

2.6.6 Input/Output Paths

The operation of the cellular automata depends on the method of protein introduction and extraction. In the case of this cellular automaton, the protein identification system is boundary-based. This boundary-based input and output is the introduction and detection of proteins only at the bounding edges of the biological film. By situating the sensor system so that the protein input channels and the protein sensing system on the physical boundary, the system can be much less complex than requiring multitudes of leads into the film's core. Furthermore, the likelihood of cellular disturbances at the outer edges is less likely than variations within the media.

In certain cases, depending on algorithm design, the boundary-based system can be replaced with an optical output method, as provided by the set of bio-luminescent *lux* sequences. This method allows sensors to detect film state based on the production, and intensity, of the bacterias' luminance. However, this output method has drawbacks in the requirements for luminescent *lux* operation that are difficult to obtain in small cell groupings [16].

2.6.7 Algorithm Design

Algorithm design for the cellular automaton requires extensive fault tolerance. Even with the multiple layers of error correction, errors will occur and the process of error correction requires time to adjust conditions to correct these variances. Therefore, the algorithms must account for these possible variances in its design. The example used above, Conway's "Game of Life," is a perfect example of fault tolerant programming for the automaton. Algorithms, for successful application, should have transition cases with similar requirements. A cell that has the state transitions that are between values, as in "If three or four neighbors are on then change state" will be resistant to critical errors, unlike a transition of the form "If the left cell is *ON* and the upper cell is *OFF* take the state of the cell to the right."

Chapter 3

Experimentation and Results

The goal of the experimentation is to demonstrate the self-tiling nature of the system, the accuracy of the genetic rules, the properties of the error correction systems and to reveal any limitations that the system may have. Testing on a true biological system is possible, but because of the advanced nature of this experiment, the necessary genetic modifications would require longer than this investigation allows. It is feasible to produce a simulation of reasonable accuracy that allows us to test the correctness of the system design.

3.1 Software Selection

Several packages are available for simulating cellular automata as there are several packages available for simulating complex biological processes. However, there are no packages that do both functions simultaneously with the features this experiment required, such as probable cell die-off and the differentiation between intercellular and intracellular proteins. Therefore, it was

necessary to design software to accurately simulate the bacterial film as a cellular automaton while staying true to the biological aspects of the cellular systems. The software was designed using Sun's Java^{TM}language. While other languages, such as C or Fortran would have faster simulation times and possibly be more memory efficient, the rapid software generation cycle and ease of generating a graphical interface were the primary reasons this language was selected.

3.2 Setup of Experiment

The experiment is modeled after a common simulation of a four-neighborhood cellular automaton. The simulation is over a two-dimensional grid, N by N units in size where N is an even integer. The initial tiling for this experiment is regular, with each cell that is not an environment boundary cell having exactly four neighbors. Each cell is represented by a Java Object possessing six significant features:

- Lists of present intracellular and intercellular proteins
- Pointers to cells in the four-neighborhood
- An array listing active genes
- A rule set that defines each gene's operation
- The living or non-living status of the cell
- A variable defining the time until the cell is processed.

The cells on the boundary of the grid are inactive placeholders, providing a region for chemical input and simplifying the design of the simulation.

3.2.1 System Parameters

There are five system parameters that alter the process of system initialization. The first parameter is the random selection function. Each cell's time until processing a variable is initialized with a uniformly distributed random integer between a pre-defined maximum and minimum value. As each cell is reached in the simulation, that value is checked; if zero, the cell is processed and assigned a new random value. However, if the value is not zero, the cell is not processed and the value is decremented. This random variable simulates the non-synchronous operation of the bacteria and the different rates they may process their chemical messengers.

The second parameter is the die-off rate. Upon being visited, the cell has a fixed chance of dying off. If the cell dies, the cell's status is listed as dead and is current values, including protein lists, are cleared.

Related to the second parameter is the third parameter, the cell's replication rate. If a cell's state is dead, there is a chance that a new cell will take its place. Thus, the dead cell is reset to an original, uninitialized state, and then processing is bypassed, the entire cycle being consumed by the cell's replication.

The fourth parameter is the grid size. As the size of the grid increases, the number of possible interactions increases as well. The larger the grid, the more elements that can possibly die-off and the greater number of ways the random selection function can choose to select cells

during initialization.

The final parameter is the maximum iteration size. By controlling the number of passes across the grid, each tiling sequence, one-way, two-way and four-way, can be extended to observe any long term effects that might occur as a result of die-off or physical disruption of the environment. By shortening the number of passes, the effects of partial initializations can be observed.

3.3 Simulation

3.3.1 Testing Sequence

The simulator was run fifty times on a ten by ten grid. Each simulation was initialized with a different random key, providing fifty distinct test runs. The test sequences proceeded in the following order, introduction of the one-way initiator, introduction of the two-way initiator and then introduction of the four-way initiator. Between the introduction of each protein initiator the simulator runs for one-hundred iterations. During and after each tiling level's iterations, permanently fixed tiling errors and significant state re-orderings were recorded.

3.3.2 Error Introduction

There are four operations that introduce error conditions in order to test the auto-correction features of the cellular system. The first path is through the cell die-off rate in the cell's processing function which was described earlier. The final three operations that test the error-correcting features of the system are three functions that alter the four-way tiled cellular grid in specific

patterns and then allow the grid to proceed to an equilibrium.

The first function simulates a mass cellular die-off, randomly killing one in six cells and then processing the grid for one-hundred iterations. The second function simulates a minor cellular die-off, randomly killing one in twenty cells and again processing the grid. The final function causes a one in twenty chance of two neighboring cell's switching locations, simulating a structural disturbance in the bacterial film.

3.3.3 Summary

The simulations revealed three primary features of the tiling nature of the network. The first network feature is that the tiling algorithm is successful over the network with one notable error condition arising in certain trial runs. During the two-way tiling, the initialization path choice can cause a loop-back error as shown in figure 3.1. All non-boundary cells will eventually be subject to the error correction systems with only the boundary cells providing an environment where this error can remain in a fixed state. Since the four-way tiling is based on the accuracy of the two-way tiling, the error propagates into the four-way tiling. However, if this error is accounted for, or the error is manually repaired through the introduction and removal of the Ω control proteins along the effected boundary, the error is not significant.

The second network feature shown by the introduced error tests is that the network usually resets to the correct tiling state within three iteration of the error introduction. The only cases where it took significantly more iterations to repair was if a loop-back error existed along the $(0, N)$ to (N, N) set of cells and die-off or structural disturbance simulation caused the loop-

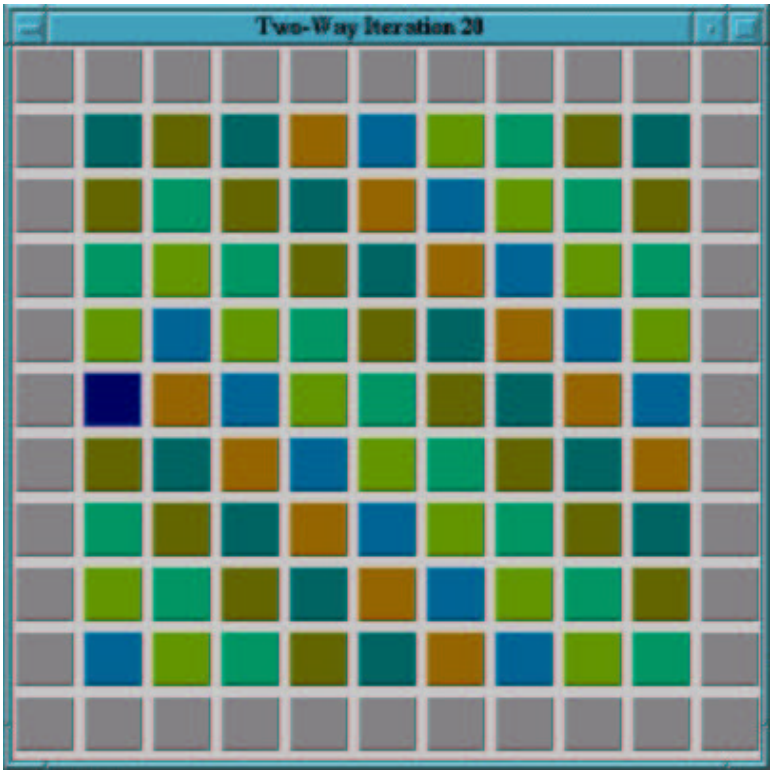

Figure 3.1: Loop-Back Error between $(0, 0)$ and $(0, N)$

back error to correct to the proper tiling.

Chapter 4

Future Work

4.1 State Reduction

In order to have a functioning bacterial automaton, this system relied on fifty distinct modified gene sequences. Altering this number of sequences is possible, but difficult to do in the same bacterium. The number of required genetic sequences must be reduced for this system to have applications in real world biocomputation. Several schemes were used to reduce the original number of over one-hundred seventy-five sequences to fifty and other similar schemes may still exist.

4.2 Simultaneous Computation

Once a tiling is established in a cellular film, it is possible to run simultaneous processes on the bacteria. Instead of having only one set of proteins signifying *ON* and *OFF* with the cor-

responding protein control sequences, it is possible to have two or more sets of gene encoded algorithms with orthogonal sets of signaling proteins functioning over the film. This promotes efficient use of the film and can be used to further promote error correction by having two versions of similar algorithms working over the network, and then comparing their answers as time progresses.

4.3 Six-way Independence

While four-way independence required a significant number of proteins and altered gene sequences, six-way independence would provide a method for exploiting a three-dimensional cellular environment for computation, increasing the computing power and overall functionality of the network. However, the complexity of six-way independence, along with the further issues of bacterial density over three dimensions will cause this issue to be complex and possibly not a likely candidate for implementation.

Bibliography

Bibliography

[1] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning ways for your mathematical plays*, volume 2. Academic Press, New York, NY, USA, 1982.

[2] Arthur W. Burks. *Essays on Cellular Automata*. University of Illinois Press, Urbana, IL, USA, 1970.

[3] T. Gardner, C. Cantor, and J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403: 339–342, 2000.

[4] Genetech Inc. Induction of the Lac Operon. Revised January 2, 1999. Retrieved November 28, 2002 from http://members.tripod.com/geneticweb/induction.html.

[5] Thomas H. LaBean, Erik Winfree, and John H. Reif. Experimental progress in computation by self-assembly of DNA tilings. In *DNA Based Computers V*, 1999.

[6] M. Leonard. Adleman: Molecular computation of solutions to combinatorial problems. *Science*, 266: 1021–1024, November 1994.

[7] Richard J. Lipton. DNA solution of hard computational problems. *Science*, 268: 542–545, 28, 1995.

[8] Norman Margolus and Tommaso Toffoli. Cellular automata machines. *COMPSYSTS: Complex Systems*, 1, 1987.

[9] V.I. Merkulov, M.A. Guillom, D. H. Lowndes, M. L. Simpson, and E. Voelkl. Shaping carbon nanostructures by controlling the synthesis process. *Applied Physics Letters*, 79:1177–1180, August 2001.

[10] K. Quon, B. Yang, I. Domian, L. Shapiro, and G. Marczynski. Negative control of bacterial DNA replication by an essential two-component signal transduction protein. *Cell*, 84, 1996.

[11] Diana Rooss and Klaus W. Wagner. On the power of DNA computing. *Information and Computation*, 131: 95–109, 1996.

[12] J. A. Rose, Y. Gao, M. Garzon, and R. C. Murphy. DNA implementation of finite-state machines. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 479–490, Stanford University, CA, USA, 1997.

[13] S. Ulam. *A collection of mathematical problems*. Interscience, New York, USA, 1960.

[14] David Ussery. Central Dogma of Molecular Biology. Revised September 8, 1998. Retrieved November 28, 2002 from http://www.cbs.dtu.dk/dave/DNA_CenDog.html.

[15] John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, USA, 1966.

[16] R. Weiss and T. F. Knight Jr. Engineered communications for microbial robotics. In *Proceedings 6th DIMACS Workshop on DNA Based Computers, held at the University of Leiden, Leiden, The Netherlands, 13 - 17 June 2000*, pages 5–19.

[17] Ying Xu and Edward C. Uberbacher. Automated gene identification in large-scale genomic sequences. *Journal of Computational Biology*, 4(3): 325–338, 1997.

Vita

Michael Campfield was born in Johnson City, NY on September 2, 1976. He was raised in Apalachin, NY and went to Vestal Senior high school. Michael moved to Knoxville, TN in 1994 and pursued a degree in Chemical Engineering. Following his switch to Computer Science in 1998, he received his bachelors degree in that subject in 1999, developing an interest in artificial intelligence. Michael is currently seeking the title: Master of Computer Science.