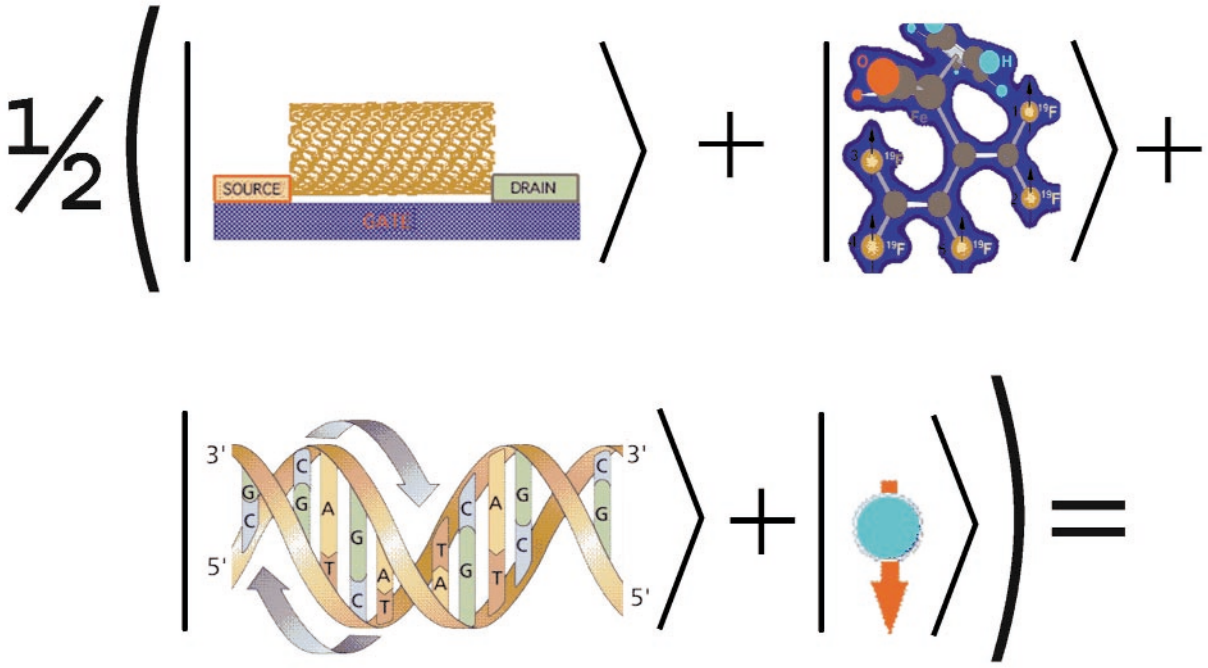


CBS^3

Computing Beyond Silicon Summer School

CALIFORNIA INSTITUTE OF TECHNOLOGY

Pasadena, California

June 17 - July 17 2002

Coordinated by Caltech Professors André DeHon and Erik Winfree

www.cs.caltech.edu/cbsss/

CONTENTS

The Computing Beyond Silicon Summer School gratefully acknowledges the generous financial support of the Gordon and Betty Moore Presidential Discovery Fund at Caltech and the Charles Lee Powell Foundation.

The faculty of the Computing Beyond Silicon Summer School also would like to acknowledge the organizational, administrative, advisory, and design efforts of Maria Katsas, Peter Mendenhall, Richard Murray, and Jessica Stephens.

Computing Beyond Silicon Summer School 2002 logo was conceived and designed by students Murali Kota, Ming-Shr Lin, David Lee, and Samuel Klein.

There is [Still] Plenty of Room at the Bottom

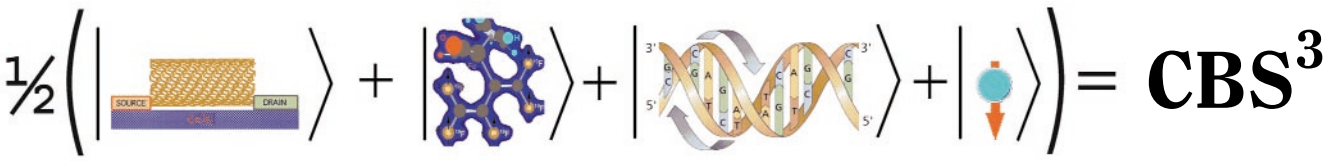
Richard P. Feynman, December 1959

INTRODUCTION

There are revolutions at hand in the way we understand and implement computation, driven by an awareness of impending barriers to VLSI scaling and new understandings of the physical world. This fundamental shift in perspective allows us to contemplate engineering computational substrates at the molecular and atomic levels. To develop and exploit these new substrates will require an intimate understanding of both the physical substrates and the nature of computation, as well as the relation between them. Research and researchers whose competency spans across the disciplines will be necessary to drive progress in this area of novel computational substrates.

In this summer program we assembled a unique collection of materials intended to start identifying the fundamental background knowledge necessary to work in this area and the fundamental challenges this area presents. We invited promising, motivated students with an interest in novel computation to join researchers from Caltech and leaders from institutions across the country for an intensive, four week introduction to this exciting area. 45 undergraduate and graduate students with diverse backgrounds in computer science, electrical engineering, biology, chemistry, or physics participated in the program.

To engage the students beyond the lectures, we asked them to self-organize into small project teams to expand on issues related to or motivated by the subject matter presented in lectures. The students had roughly three weeks to focus in on a topic and put together a brief report. This volume is a collection of the student reports. Almost none of the students were “experts” in the issues they studied when they entered the program. Nonethe-less, these reports show the multi-disciplinary teams they assembled were able to dig deeply into a number of interesting problems and point out some promising directions for further inquiry.



GUESTS LECTURERS

Len Adleman
USC, Professor of Computer Science and Molecular Biology

Dave Bacon
Caltech, Institute for Quantum Information (IQI) Postdoctoral Scholar

André DeHon
Caltech, Assistant Professor of Computer Science

Michael Elowitz
Rockefeller University, Postdoctoral Fellow at the Center for Studies in Physics and Biology

James R. Heath
UCLA, Professor of Chemistry and Biochemistry

Tom F. Knight
MIT, Senior Research Scientist at the MIT Artificial Intelligence Laboratory

Phil Kuekes
Hewlett Packard, Quantum Science Research

Hideo Mabuchi
Caltech, Associate Professor of Physics

Marcelo Magnasco
Rockefeller University, Associate Professor of Mathematical Physics

Norm Margolus
Research Affiliate at the MIT Artificial Intelligence Laboratory, Research Professor at the Boston University Center for Computational Science

Ashwin Nayak
Caltech, Institute for Quantum Information (IQI) Postdoctoral Scholar

Nicholas Pippenger
University of British Columbia, Professor of Computer Science

Leonard J. Schulman
Caltech, Associate Professor of Computer Science

Ron Weiss
Princeton, Assistant Professor of Electrical Engineering

Erik Winfree
Caltech, Assistant Professor of Computer Science and Computation and Neural Systems

Algorithmic Self-Assembly of Circuits

Michael deLorimier Alexandre Mathy Dustin Reishus Rolfe Schmidt
Bilal Shaw Li Chin Wong

August 24, 2002

Abstract

1 Introduction

The order we see in nature is self-assembled: atoms, crystals, cells, animals, solar systems, and galaxies are just a few examples. While we have been able to build phenomenal devices with controlled assembly techniques, we still cannot engineer machines at the scale of the ribosome. Can we understand the mechanism behind natural self-assembly and harness it to fabricate new devices? A systematic attack on this problem as been initiated [5, 1], and in this paper we use these techniques to explore how self-assembly could be used to fabricate nanoscale electrical circuits.

We start with a simple model, where molecules are thought of as two dimensional square tiles [Wang] having different glue types on north, east, south and west sides. These glues have strengths associated with them which will be important later on. Now tiles that have matching glues stick to each other. The model that we have considered is an irreversible one, which means that once tiles stick together with a certain bond strength they do not come apart.

In reality, molecular bonds are reversible. But some bonds are stronger than others, and if a molecule is attached in more than one place it is less likely to fall off. Analysis of the thermodynamics of these interactions shows that for some temperatures our model that "tiles with matching glues stick" is actually fairly accurate [7]. This is called the $T = 1$ model: the temperature T is such that only one matching bond is needed for two tiles to stick to each other. The behavior is different at other temperatures. As the temperature rises above $T = 1$, eventually it reaches a point where in equilibrium, a tile sticks to other tiles only if it can find two bonds (or one double strength bond). This gives us a new model of self assembly which seems more powerful than the first. Not surprisingly, it is called the $T = 2$ model.

In this paper our purpose is to assemble some interesting circuit with the $T = 2$ model of self assembly. One could imagine fabricating tiles with nano dimensions which have logical gates embedded on their surface. Then one need only program these gates in the appropriate way to assemble the tiles into meaningful circuit. For our presentation we chose to assemble the Fast Fourier Transform (FFT), but the ideas we use can be easily modified to assemble sorting networks, power-law crossbars, and various decoders.

Designing a tile system to assemble a network is a programming task, but the program is massively parallel and not at all like programs we are used to writing for machines with limited numbers of processing nodes and high context switch overhead. However, when writing a program for a multi-processor machine it is possible (and sometimes best) to just use one processor. Similarly, when designing a tile system we can make it emulate a "serial" program to simplify our task. We do this by taking advantage of the fact that every bounded CA rule with the Margolus neighborhood can easily translated into a $T = 2$ tile system. The glues in this tile system correspond to symbols in the CA rule, and tiles correspond to rules. When this tile system is seeded with "input data", it will assemble the trace of the CA's evolu-

tion on this input.

CA are easy to think about, and we take advantage of this to design a CA that produces the recursive shape we need for the FFT network. Once this is done, we translate the CA rules into a tile system that uniquely assembles our network.

The sequel proceeds as follows. In Section 2 we provide a brief overview of the uses and implementation of the Fourier transform. Section 3 describes how we designed CA rules to produce the recursive FFT shape, and follows with a discussion of the actual $T = 2$ tile system. Space does not allow a full presentation of the rules, but this is available online [4]. With the tile set in hand, we analyze the (in)tractability of implementing this system with DNA DX molecule tiles in Section 4. Finally, in Section 5 we analyze the error rates of systems like ours and propose a way to reduce the rate of failure do to tile misincorporation.

2 The Fast Fourier Transform

For all Abelian groups G, the Fourier Transform is the unitary change of basis on $L^2(G)$ that simultaneously diagonalizes all translation invariant operators. This transform plays a pivotal rôle in pure mathematics. For continuous groups, differential operators are translation invariant, and differential equations become algebra problems. Convolutions are just translation invariant "moving averages", and become pointwise multiplications. The applications are not limited to differential and integral equations: when $G = (\mathbb{R}^+, \cdot)$, and $\omega = \sum_{\mathbb{N}} \delta_n$ is the Dirac comb along the natural numbers, the Fourier transform of ω is just the Riemann zeta function. Understanding the zeroes of this function would allow us to place strong bounds on the distribution of the prime numbers.

As important as the Fourier transform is in pure Mathematics, most people who use it do so for very practical reasons: there is a fast divide-and-conquer algorithm, called the Fast Fourier Transform [FFT], for computing the transform. This makes it feasible to evaluate convolutions and solve many differential equations very quickly by transforming the problem to the Fourier domain, solving the easier problem, then performing the inverse FFT to bring the solution back. This approach is used widely, and in applications like signal processing speed is so crucial that is economical to implement the FFT in hardware.

How does the FFT work? On the group $G = \mathbb{Z}^N$ the Fourier transform of a function f can be written

$$\hat{f}(\xi) = \sum_{x \in \mathbb{Z}^N} f(x) \exp[-2\pi i x \xi / N]$$

Notice that

$$\hat{f}(\xi) = \hat{f}_{\text{even}}(\xi) + e^{-2\pi i \xi / N} \hat{f}_{\text{odd}}(\xi)$$

when $\xi < N/2$ and

$$\hat{f}(\xi) = \hat{f}_{\text{even}}(\xi - N/2) - e^{-2\pi i \xi / N} \hat{f}_{\text{odd}}(\xi - N/2)$$

when $\xi >= N/2$.

So we can evaluate this recursively. By the Master theorem [2], the running time of this algorithm is

$$T(N) = N + 2T(N/2) = \Theta(N \lg N)$$

which is much faster than the naive $\Theta(N^2)$ approach.

This algorithm can be implemented as a network of phase-shifters and adders. If we were given "magic boxes" that compute $\hat{f}_{\text{even}}$ and $\hat{f}_{\text{odd}}$, then it would be easy to build a network for the full Fourier transform (see Figure 1).

But we do not need magic- each of these boxes is simply implementing a smaller FFT, so we can continue building the network as seen in Figure 2.

There is one problem with this network as presented. The FFT requires us to keep splitting the array into even and odd subarrays, but our network splits it into high and low subarrays. In other words, we are checking the high bit of the array offset when we should be checking the low bit. We can fix this by performing a bit reversal permutation on the array before passing the array to the FFT network.

3 CA Rules for the FFT Network

To self-assemble a circuit that performs the fast fourier transform, we envision self-assembly from

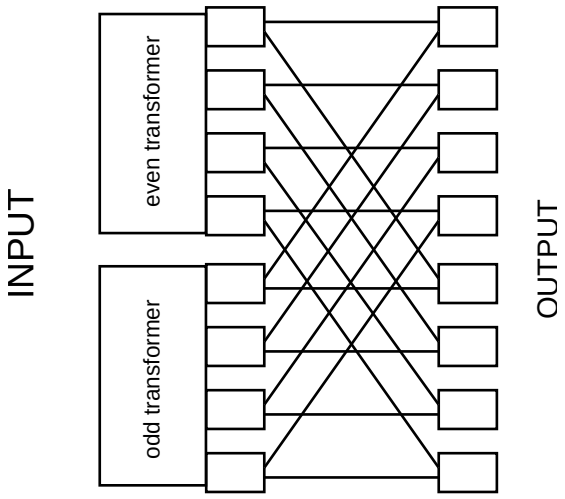


Figure 1: Implementing an FFT network with magic boxes

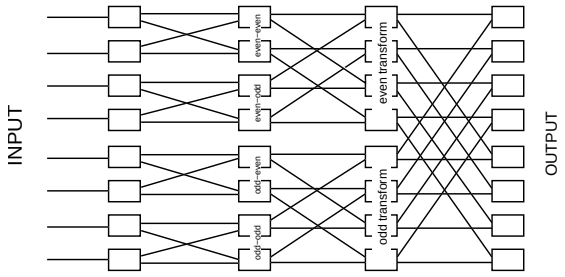


Figure 2: The FFT network

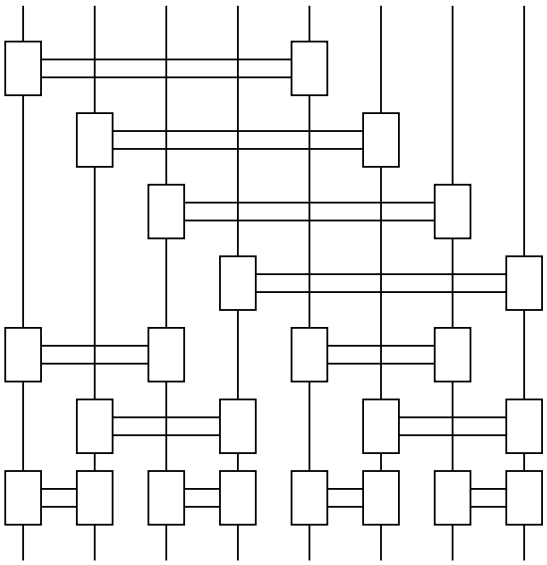


Figure 3: Self-Assembled FFT Network

a cellular automata perspective. Briefly, the T=2 model of self assembly can be thought of as a 1-dimensional cellular automata using the Margolus neighborhood. The specifics of how to transform 1D CA rules into a T=2 tile set will be covered in more detail in the next section. Thinking of self-assembly from a CA standpoint allows us to design self-assembled circuits in terms of particles and collisions, which are CA symbols and interactions between symbols, respectively.

An FFT network consists of many small logic blocks interconnected in a very specific way. The logic blocks perform mulitplications, and the interconnects tell them which signals to multiply. The structure of the circuit is highly self-similar; it is this recursive structure that led us to believe that the FFT network was a good candidate for production by self-assembly. In our design of the completed circuit, we have interconnects travelling horizontally and vertically through every tile. At certain locations along diagnals, the horizontal and vertical busses are connected at a logic block. See Figure 3 below for a schematic diagram of the logic blocks and interconnects.

Our self-assembled circuit consists of tiles that line up the interconnects and places the logic blocks in the correct positions. To locate the places where a logic block should be, we use a system of particles and collisions. As we mentioned before, the particles are symbols in a 1D CA using the Margolus neighborhood. The Margolus neighborhood is a way of implementing reversible CA rules. It consists of updating the CA in two phases: In the first phase, an even numbered cell looks at its neighbor to the left, an odd numbered cell looks at its neighbor to the right, and every cell updates based on its symbol and its neighbor's symbol. In the second phase, each cell looks at its other neighbor.

We created CA rules for four kinds of particle movement: stationary, left and right unit speed, and left moving half speed. Stationary particles are called μ, left moving particles are called λ, right moving particles are called ρ, and left moving half speed particles are called γ. The CA rules to implement stationary particles and particles that move with speed one are trivial. A half speed left moving particle's CA rules are slightly more complex. It completes one cycle every four phases of the CA and moves two cells per cycle. The first two phases it moves with speed one, and the next two phases it is stationary, after which it repeats this procedure. The movement must be two cells every four phases, because if it moved only one cell in two phases it would be in the wrong neighborhood and unable to move the following phase.

As you can see from the collision map in Figure 4, the FFT circuit has a recursive structure. At the top (time t_0), the only particles are the left and right boundry particles and a stationary particle in the middle. These immediatly eject ρ, λ, and γ particles. When the ρ and λ collide in the center, they create a μ and keep travelling. The λ collides with the left boundry, the ρ collides with the right boundry, and the γ collides with the μ at time t_1. This completes the first section of the FFT network. Here the recursion enters: The right and left boundries eject ρ and λ particles and the μ ejects ρ, λ and γ particles and serves as both the left boundry and the right boundry for the two halves of the circuit. This process continues until time $t_{\log(n)}$, when every particle is a μ. If logic blocks are placed on every ρ tile, the FFT network is complete.

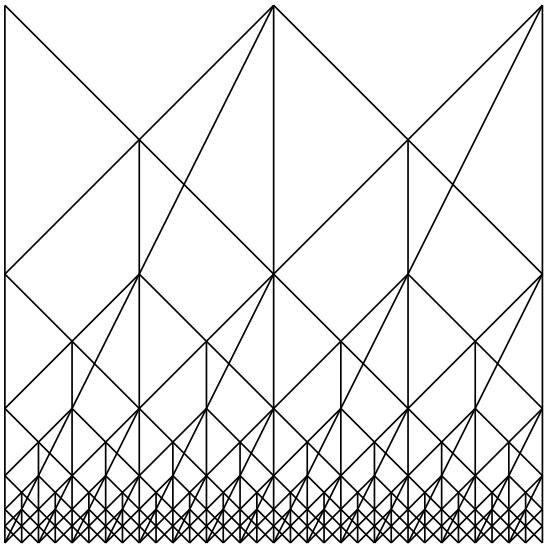


Figure 4: Collision Map

There are several difficulties when creating the rules for the CA with the Margolus neighborhood. First, the λ and ρ cannot start in the same phase. The neighborhood adds constraints so that the particles cannot always move in a particular direction; sometimes they must wait one phase before moving. An additional difficulty with creating CA rules for particle movement is that the number of CA symbols required typically grows as a power of the number of logical particles. This is compounded by the fact that the number of explicit rules can grow as a power of the number of rules.

This leads to a large number of explicit rules for a relatively small number of logical particles. This ends up being very bad for the implementation of any self-assembled circuit, because the each rule corresponds to exactly one tile. This is how the CA model, in which we designed our FFT network, can get translated to a $T = 2$ tile set. Along the boundries and on the input row, tiles with double bonds are used. For every rule in the CA, one tile is created in the tile set.

4 Implementation Issues

We explored the feasibility of experimentally building the self-assembled Fast Fourier Transform using DNA tiles. Suppose we use the double crossover (DX) molecule [6] to implement our one tile-one CA rule and one bond-one CA tile self-assembly system. The four sticky ends of the DX molecule will represent the north, south, east and west glues of the DNA tile. The south and west glue will represent the input and the north and east glue will represent the output.

To design the sequences of the unique sticky ends, we need to consider the optimal length and the corresponding sequence complexity. The number of tiles required to build the network, excluding the seed tiles, will be 73. If glue A exists in the north position, the complementary sequence will represent the A-south glue. A total of 2 different sequences will be required to represent this glue. Otherwise, if a glue occurs in both the north/south and east-west position, a total of 4 different sequences will be required to represent the glue. An analysis of the CA rules that showed that there are 45 symbols that occur exclusively in the north/south or east/ west position, and 24 symbols occur in both north/south and east/west position. Therefore a total of 69 different glues and the corresponding 69 complementary strands are required to build the experimental tile set. Assuming that we use the same sequence for all the non-sticky end portions of the DX molecule, we will be synthesizing 140 (2 + 69 + 69) different strands of DNA.

Assuming that we use sticky-ends which are 6nt long. The sequence space that can possibly be explored is 4096. However, in an attempt to control the hybridization between strands, the GC-content of each strand is kept constant at 50%. This reduces the sequence space to 1280. The sequence design should also attempt to avoid formation of undesired secondary structures and to maximize the interstrand interactions to form a stable double crossover structure. The 6nt sticky-ends should also not be complementary to the tile core.

Potential technical problems that will be faced include the problem of stoichiometry. According to the

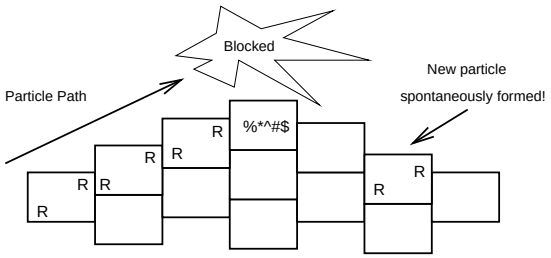


Figure 5: Errors that can occur in CA rule assembly

$T = 2$ model, concentration of different strands will be kept constant during self-assembly. However, as self-assembly occurs experimentally, the concentration of molecules that are being self-assembled decreases. This problem is even more pronounced if one sticky-end is used heavily compared to other sticky ends. Another example of a technical problem will be finding the optimal temperature for optimal self-assembly. Different sticky-ends might interact optimally at different temperatures. This problem is partially solved by keeping the GC-content constant.

As shown in the analysis, implementing the self-assembly of the FFT circuit using DNA tiles is not an easy task, although not entirely impossible.

5 Error Correction

Our tile sets depend critically on particle scattering CA rules. Unfortunately, assembly errors do occur and the tile sets we have designed are particularly sensitive to these errors:

- If a particle tile gets misincorporated it will propogate and be locked in quickly.
- If a misincorporation occurs on a particle path it destroys the particle.

To see this more quantitatively, consider a tile system that implements a CA rule with one moving particle in open space that has a misincorporation rate of ϵ. It is straightforward to show that in an assembly of area A

- the expected number of spontaneous wires is $\epsilon A + o(\epsilon)$

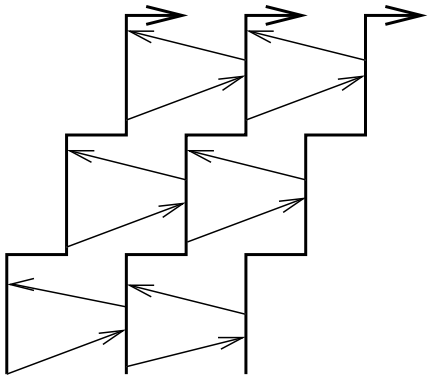


Figure 6: A self monitoring particle team

- the expected path length is $\frac{1-\epsilon}{\epsilon}$

Thus if 1 out of every tiles is a misincorporation (which would be an excellent error rate in a system with 73 tiles), we can only expect particle paths of length 99.

We would like to replace this with something more robust, so we propose using a "team" of particles that monitors and repairs itself. Figure 6 shows a schematic view of how such a system could work- three particles move in synchrony, sending signals to their neighbors. A set of CA rules that implements this strategy to repair all single errors in the signal path and prevent all single errors from creating a spontaneous particle is available at http://www-scf.usc.edu/ rolfesch/teamspirit.html. This set of rules fully repairs all spatially isolated errors within a constant number of time steps, but may fail to correct multiple errors. Thus it improves the expected particle path length to $\Omega(\epsilon^{-2})$ and reduces the number of spontaneous particles in an assembly of area A to $O(\epsilon^2)$.

However, this set of rules is enormous compared to the initial two free-particle rules, particularly if wildcards need to be expanded. We note that the rules were designed every symbol except for 0 has a unique left and right neighbor, so a misincorporation cannot lead to a valid rule application at the next step. This prevents erroneous tiles from being locked in quickly, and may allow us to implement our wildcard rules with tiles that simply have no glue on the wildcard side. These wildcard tiles will have no particular advantage over competing tiles when attaching to an erroneous assembly, but once attached they can be locked in by valid rule applications immediately.

In principle, this mechanism can be extended to teams of k particles that either simulate an error-correcting CA rule like GKL [3], or implement a voting protocol to determine whether the path should continue or terminate. The number of tiles required for such a system would be constant for the GKL rule, and polynomial in k for the voting rule. This sounds reasonable, but in practice the tile sets are unreasonably large. As we noted above, asymptotics can be falsely reassuring when designing DNA tiles for self-assembly. The constants are critical, and the absolute number of tiles must be small.

6 Conclusion

Our work has shown us that it is fairly simple to design tile systems that assemble common networks with $O(1)$ tiles. Unfortunately we also realize that it is very difficult to assemble common networks with a reasonable number of tiles. We relied heavily on CA rules to design our tile systems, and it would be interesting to know how much efficiency we give up with this approach. The CA rule approach also led us to a simple error correction scheme, which seems appealing in principle, but uses far too many tiles to be practical. In short, we have answered the querstions we set out to answer, but the standards we were using to measure success seem too liberal. We now realize that the real problems are much more challenging.

References

[1] L. Adleman. Toward a mathematical theory of self-assembly, January 2000.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2002.

[3] P. Gacs, G. Kurdiumov, and L. Levin. One-dimensional homogeneous media dissolving fi-

nite islands. *Problemy Peredachi Informatsii*, 14(3):223–226, 1978.

[4] http://www-scf.usc.edu/ rolfesch/cbsss.html.

[5] P. Rothemund and E. Winfree. Theprogram size complexity of self-assembled squares. In *STOC*, 2000.

[6] E. Winfree. Ph.d. Thesis, Caltech, 1998.

[7] E. Winfree. Simulations of computing by self-assembly. In *Proceedings of the 4th DIMACS Meeting on DNA Based Computers*, June 1998.

Artificial Life Through Evolutionary Computation

California Institute of Technology
Computing Beyond Silicon Summer School
August 17, 2002

Bryan Blumenkopf
Massachusetts Institute of Technology
bblumen@mit.edu

Ashley Holtgraver
Carnegie Mellon University
ashleyh@andrew.cmu.edu

INTRODUCTION

The following is a summary of work prepared over a two-week period as part of the Computing Beyond Silicon Summer School, offered by the California Institute of Technology during the summer of 2002. The objective was to prepare a pedagogical lecture on large-scale artificial life as an outgrowth of methods in the computational simulation of evolution, aimed toward an audience of graduate students and advanced undergraduate students in biology, chemistry, physics, engineering, and computer science.

OVERVIEW

Artificial life is loosely defined as the computational simulation of biological systems, both small (cells and cell networks, genetics, etc.) and large (organisms, populations, evolution). Here we focus on large-scale artificial life, in particular the modeling of virtual creatures through evolutionary methods.

Though other approaches are certainly possible, a powerful and well-studied method of simulating large-scale biological systems is *evolutionary computation*, drawing inspiration from Darwinian natural selection. Evolutionary computation simulates natural selection algorithmically, automatically "evolving" optimal solutions according to a prespecified measure of fitness. In particular we discuss the genetic algorithm as the primary basis of evolutionary computation.

Lastly, we show one method of evolving virtual organisms by summarizing the work of Karl Sims in his landmark 1994 publication on the subject.

EVOLUTIONARY COMPUTATION

Evolutionary computation (EC) is a general term for several computation techniques which are all based to some degree on the development of biological life in the natural world. Currently there exist several major evolutionary models.

The *genetic algorithm* (GA), by far the most common application of EC, is a model of machine learning taking inspiration from genetics and natural selection. In natural evolution, each species searches for beneficial adaptations (species optimizations), which arise through mutation and the chromosomal exchange and recombination of breeding. The two key axioms underlying the genetic algorithm are that complex nonbiological structures can be described by simple bit strings (analogous to the "genetic code" of chromosomes), and that these strings could be improved, according to a particular measure of fitness, by the application of simple transformation functions (just as living species may be "improved" through mating). This will be described in detail shortly.

Evolutionary strategies (ESs) simulate natural evolution similarly to the genetic algorithm. Like GAs, ESs are most powerful

while comparing populations of data, as opposed to individual samplings. Differences between the two lie in their application; ESs were designed to be applied to continuous parameter optimization problems seen in laboratory work, while the GA was used originally in integer optimization problems.

Evolutionary programming (EP) is a stochastic optimization function, similar in many ways to the GA. However, EP places emphasis on the behavioral link between parent and offspring, as opposed to the GA's attempt to model the exact code transition as seen in nature. EP follows a general process with obvious similarities to natural evolutionary progression. An initial population of trial solutions is selected at random from a coding scheme. A chosen mutation factor is applied to each solution, generating a new population. Because EP resembles biological evolution at the level of reproductive populations of species, and there is no genetic recombination between species, EP's transformations take place without *crossover*- combination of two parent member's genetic code. The offspring species' members are weighed for overall fitness; the best are kept while the rest are culled, and the algorithm repeats with the new, more fit population.

The *learning classifier system's* (LCS) purpose is to take in input and produce an output representing a classification of that input. They have undergone and continue to go through multiple minor changes of name and scope, but the enduring foundation originates in J.H. Holland's Adaptation In Natural and Artificial Systems, wherein he envisions a cognitive system capable of classifying and reacting appropriately to the events in its corresponding environment. This most obviously parallels the inherently intelligent behavior seen in all macro- and microscopic living creatures.

Though there are certainly other forms of evolutionary computation, the above offers a brief summary of the most established and useful evolutionary techniques. The genetic algorithm in particular lends itself well to the macroscopic forms of artificial life.

GENETIC ALGORITHMS

Genetic algorithms use the evolutionary process of natural selection as a metaphor for what is essentially a hill-climbing search without backup. GAs search for an optimum or global maximum among what is typically an enormous set of data by computationally modeling the alteration, recombination, and propagation of genes that forms the basis of biological evolution. To achieve this, certain complex biological details of evolution must be abstracted in favor of more relevant principles. Thus GAs share the following properties:

- GAs process an encoding of relevant parameters, typically represented as a DNA-like string.

- GAs search among a population (typically large) of individual state nodes.

- The optimality of each state node is evaluated according to a specified *fitness function*. This function determines, probabilistically, which nodes are propagated.

- Node propagation ("reproduction") is nondeterministic. This discourages the GA from mistaking local maxima for global maxima. Genetic splitting and pairing, as well as phenomena such as *crossover* and *mutation*, are modeled probabilistically.

Assuming that parameter encoding, population size, propagation iterations, genetic operators, and a fitness function have been chosen, the genetic algorithm proceeds as follows:

1) A population of given size is initialized.
2) For a specified number of generations:
   a) Assign each individual node a fitness level according to the fitness function.
   b) Probabilistically select a specified number of pairs of individuals according to fitness levels. Higher fitness levels increase an

individual's chance of being selected.

c) Apply the specified genetic operators to these chosen pairs to produce new individuals.

d) Randomly select individuals from the population. Replace them with the newly produced individuals.

3) Return the individual with the highest fitness level as the output.

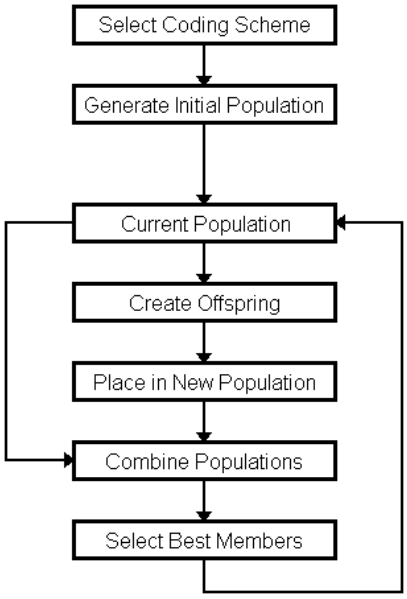


Figure 1. Graphical outline of the genetic algorithm.
From http://members.aol.com/btluke/evprog.htm

A careful choice of genetic operators can improve the efficiency of the genetic algorithm or enable it to find otherwise inaccessible solutions.

Crossover switches two subsequences of two parent strings; the goal is to place two fit sequences on the same string. Subsequences are selected probabilistically.

124423 | 424314224 331412424314224
- - >
331412 | 311132314 124423311132314

Mutation introduces "genetic diversity" into the population by randomly altering one character of an individual string. Mutation provides a way to help the GA avoid the situation in which the system fixates on a local maximum after repeatedly propagating a particular character.

1441**2**3142132 - - > 1441**4**3142132

Inversion reverses the order of a particular subsequence.

314**121313**3141 - - > 314**313121**3141

EVOLVING VIRTUAL CREATURES

An important means of simulating the evolutionary development of large-scale organisms was described by Karl Sims in the landmark 1994 publication, "*Evolving Virtual Creatures*." Sims employed genetic algorithms to traverse a directed graph bound by the constraints of a modeled physical world and ultimately create a physically feasible computational creature optimally designed to accomplish a particular task; the following briefly summarizes the premise of his work.

Each node in Sims's directed graph represents a specific physical part – such as a head, arm, or hand -- in an analogous 3D physical structure corresponding to a particular graph traversal *(see Figure 2)*. A virtual "creature" is generated by traversing the directed graph; recursion limits and constraints on the placement and physicality of parts are defined by variables intrinsic to each node and internodal connection.

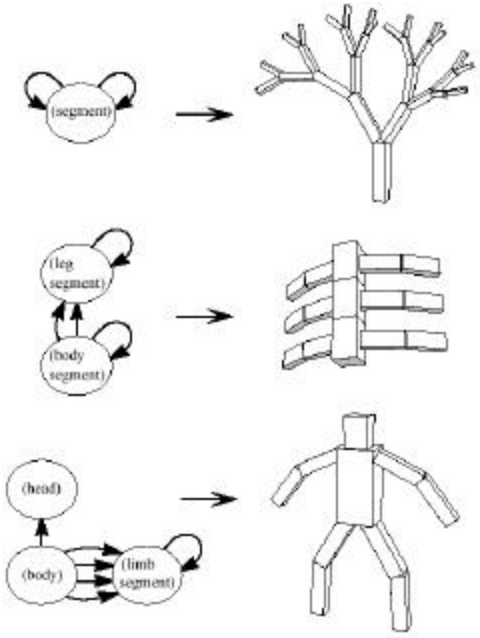


Figure 2. Correspondence between directed graph and 3D morphology in Sims's study. From Karl Sims, Evolving Virtual Creatures.

Behavior of the creatures in this simulation is controlled by a dynamic "brain," accepting sensor values (such as joint angles, surface contact, or luminance values) from the physical environment and from virtual sensors on the morphological parts and outputting effector values as forces or torques applied to specific parts of a creature. These sensor and effector values are represented as positive or negative scalar quantities.

Contained within each graph node are internal "neural nodes," assembled as an artificial neural network. This "neural structure" is assembled with the physical morphology of each creature. This network gives each creature a sense of internal state, enabling probabilistic or arbitrary behavior rather than the deterministic behavior that would result from purely sensory response. Figures 3 and 4 show an example of a descriptive graph and the resulting physical creature.

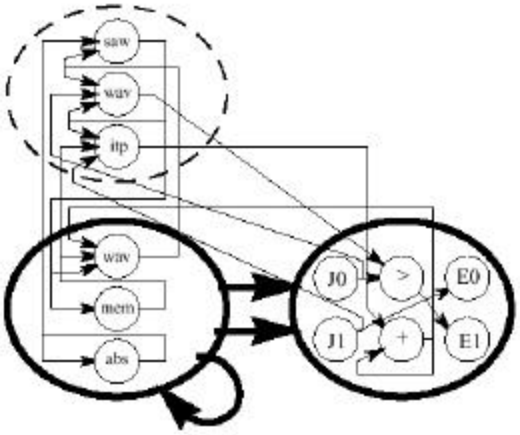


Figure 3. Example of physical structure and underlying neural structure. From Karl Sims, Evolving Virtual Creatures.

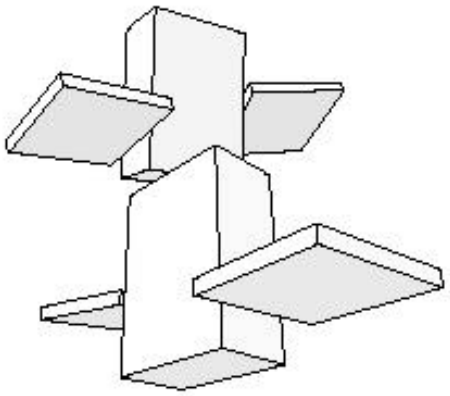


Figure 4. Corresponding physical morphology. From Karl Sims, Evolving Virtual Creatures.

The above outlines a basic description of the framework that Sims established for generating both the morphology and the control mechanisms for virtual "block creatures." An indefinite number of possible creatures exist; these creatures could be generated automatically and then "evolved" using genetic algorithm methods as described earlier. Sims placed these creatures in a simulated physical environment, and matched their performance on certain tasks – walking, jumping, swimming, following – to determine a measurement of fitness. The genetic algorithm then proceeded to mutate internal node characteristics and connection parameters, and to add or delete nodes and connections of existing creatures to create

a successive generation. Sexual reproduction was modeled by combining the nodal structures of two existing creatures. The end result of any particular iteration of this process was an automatically generated creature nearing an optimal level of fitness. Some resulting creatures of Sims's work are shown in Figures 5, 6, and 7.

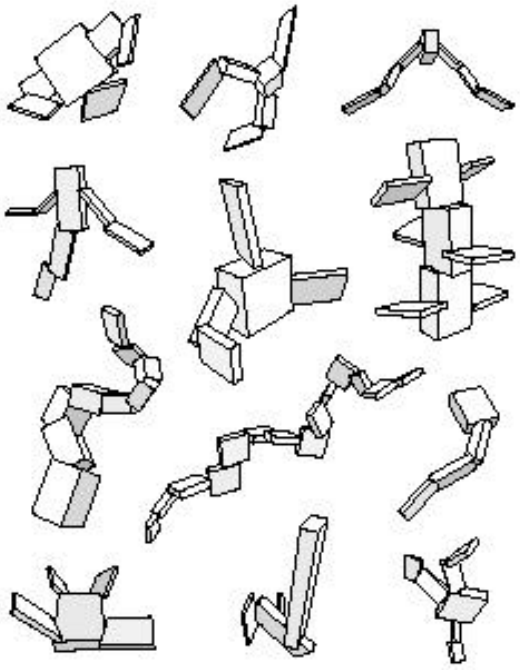


Figure 5. Example creatures evolved for optimal swimming. From Karl Sims, Evolving Virtual Creatures.

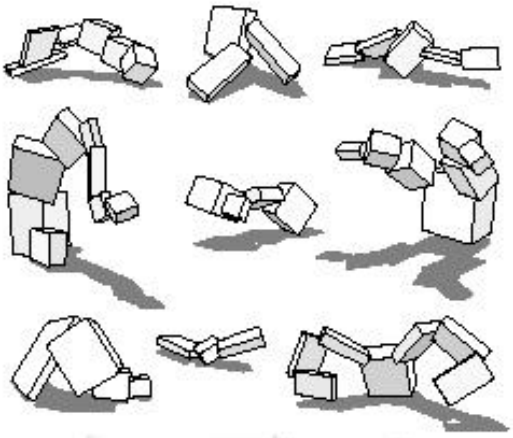


Figure 6. Example creatures evolved for optimal walking. From Karl Sims, Evolving Virtual Creatures.

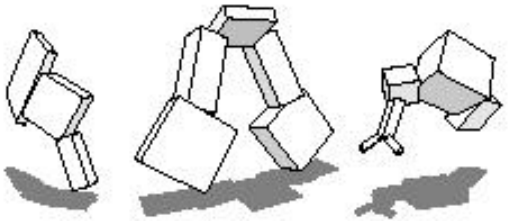


Figure 7. Example creatures evolved for optimal jumping. From Karl Sims, Evolving Virtual Creatures.

The results of Sims's work demonstrate that clever application of methods of evolutionary computation hold great potential for the autonomous generation of computational simulations of large-scale organisms.

REFERENCES

Adami, Christoph. (1998). *Introduction to Artificial Life*. New York: Springer-Verlag.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems.* Cambridge, MA: MIT Press.

Sims, Karl. *Evolving Virtual Creatures. Computer Graphics*, Annual Conference Series, (SIGGRAPH '94 Proceedings), July 1994, pp.15-24.

Sims, Karl. *Evolving 3D Morphology and Behavior by Competition. Artificial Life IV Proceedings,* ed. by R. Brooks and P. Maes, MIT Press, 1994, pp.28-39.

A Living Game of Life

Mike Blain
Ken Mizuno

The Game of Life is a set of cellular automata rules created by John Conway in the 1970s. Since this time it has been studied extensively, and is considered to be the "classic" cellular automata. This paper looks at ways to simulate the Game of Life by using living cells.

The Game of Life is played on a regular grid of cells. At each update step, if a "live" cell has one or two living neighbors, it stays alive. If a "dead" cell has exactly 3 neighbors, it is born (changes to "live"). Any other condition and the cell dies. These three rules are enough to be computationally universal. That is, the Game of Life can simulate any arbitrary Turing machine. This means that in theory, if we could simulate the Game of Life with living cells, we would have an organic computer.

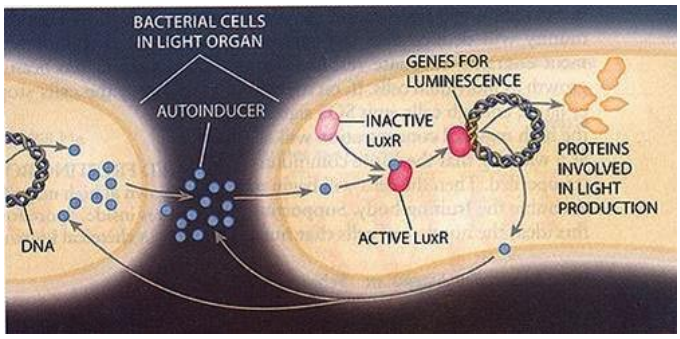
In order to play the Game of Life, a few key sub-elements have to be present. There must be a regular grid to play on, communication between neighboring cells, implementation of local rules, indication of state, and a way to set initial conditions.

The first step is creating a regular grid of cells. This grid should keep the cells in a fixed position, but allow nutrients and signaling molecules to move around freely. The simplest idea is to use plant cells. Plant cells naturally form grid-like structures with the cell walls. The downside to this method is plant cells are eukaryotes and thus it is more

complicated to add DNA information to the cells. The same problem occurs with animal cells. Through tissue engineering, it may be possible to create a regular arrangement of cells. The problem is that it would be harder to add genetic code than in prokaryote cells. The ideal cells to use for the grid would probably be bacteria. By adding plasmids to a bacteria culture, genetic information can easily be transferred to the population. The question that must be solved is then how to create a regular pattern of bacteria cells. Bacteria colonies can be grown on artificial mediums, such as agar. Bacteriostatics, chemicals which stop bacteria growth but do not kill the cells, can be used to freeze the grid once a desired size has been reached, from this point, as long as nutrients are supplied to the cells, the grid should be relatively stable. It may be the case that interesting behaviors will be present on non-regular grids. If this were true, then any bacteria colony would be sufficient, and easy to grow.

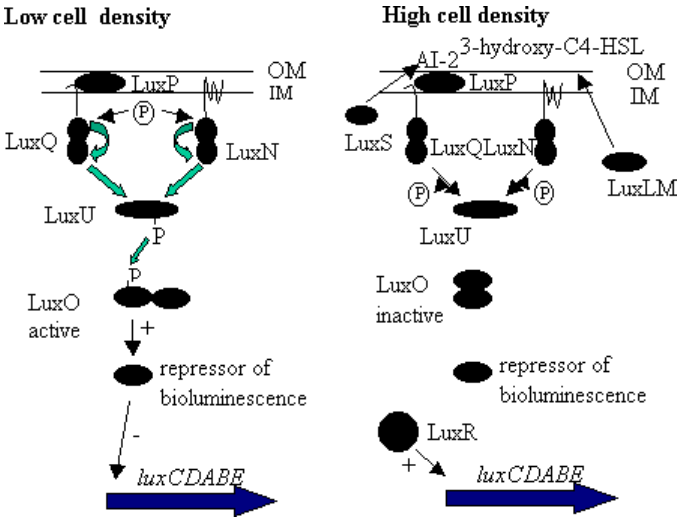
Having a nice grid structure is useless without a way to send signals between the cells. The most straightforward way of doing this would be sending signaling molecules from "living" cells that indicate this state to neighboring cells. Quorum Sensing is a common way to accomplish this in nature. Vibrio harveyi and Vibrio fischeri are both bioluminescent bacteria that glow only once a certain population density has been reached. V. harveyi and V. fisheri use slightly different mechanisms to accomplish this.

V. fischeri



In Vibrio fischeri, low levels of a homoserine lactone (AHL) signaling compound are always being produced. Once a critical population density is reached, the AHL has enough of a concentration to bind to LuxR. This activates LuxR, which promotes the production of more AHL and bioluminescent proteins.

V. harveyi



In Vibrio harveyi, the situation is almost the opposite. At low AHL concentrations, LuxO is phosphorylated and thus activated. It produces a repressor protein which inhibits production of the bioluminescent proteins. At high AHL concentrations, LuxO is dephosphorylated and the repressor is not produced, so the bioluminescent proteins are expressed.

By combining these two systems, and possibly changing which repressor or which promoter is produced, the cell could be turned on at a low concentration but turned back off at a high one. The rules for the Game of Life would have to be expressed by varying the sensitivity of these two thresholds.

The state of a given cell would be indicated by its production of the AHL and bioluminescent proteins. Glowing (and AHL signaling) indicates the cell is "alive" while not glowing and not signaling indicates the cell is "dead". This method seems much more reasonable than actually having the cells die and be born. It allows a much faster progression of steps, and more control over the game as a whole. No need to deal with mutations as generations progress, etc.

There is another problem with neighbor signaling. Ideally, corners and edges should have the same effect. But corner cells are slightly further away from the central cell. As signal diffusion is a function of distance, this may cause a problem.

The final step in simulating the Game of Life is setting an initial configuration. Since "live" cells are dependant on AHL concentrations, adding AHL to the grid would effectively set cells "on." One way to place the AHL in specific locations would be to put it in an inkjet cartridge and print the pattern onto the grid medium. Alternatively, the configuration could be pre-set on a transfer medium which is then placed on top of the grid. And the simplest method is just using a syringe to add drops of AHL where needed.

There are still some unsolved problems with all of the ideas outlined above. Very precise control over diffusion of the signaling mechanism is required. Controlling sensitivity of the signal sensing mechanisms must be precise as well. The game grid must be regularly placed identical cells, which is very hard to do with biological elements. Also, these solutions would all necessarily be implementations of an asynchronous game of life. "Clocking" the system would require yet another signaling mechanism, and probably outside control. Finally, the Game of Life itself is very sensitive to initial conditions. Small variations in initial configurations results in dramatic differences after a few generations.

The conclusions we have made are that it is not very difficult to create a game similar to Conway's, but implementing his exact rules requires a great deal of precision that may not be technically feasible. It is our opinion that interesting results, and possibly even universal computation, could be obtained from a biologically-based game of life. The elements of randomness that would be introduced to the game may actually allow more interesting games. With Conway's Game of Life, many configurations soon result in stasis or repeated patterns. An element of random diffusion, cell activity, etc would make stasis configurations and infinitely repeating patterns rarer.

Web sites consulted, and source of pictures:
http://www.wikipedia.com/wiki/Conway%2527s_Game_of_Life
http://www.nottingham.ac.uk/quorum/fischeri2.htm
http://www.nottingham.ac.uk/quorum/harveyi3.htm
http://info.bio.cmu.edu/Courses/03441/TermPapers/99TermPapers/Quorum/mechanisms.htm

Exploiting Shortcuts in Fixed Wire Programmable Network

Helia Naeimi
Dept. of CS, 256-80
California Institute of Technology
Pasadena, CA 91125
helia@cs.caltech.edu

ABSTRACT

Exploiting shortcuts in HSRA structure is considered during routing in previous jobs. It is shown in this report that, while keeping partitioning of the design fixed, by changing the placement of the design so that shortcuts are more efficiently used the path delay can be reduced. For solving this problem a greedy local algorithm and a quadrisection-based windowing method are used [1].

1. INTRODUCTION

There are number of fixed wired programmable networks for designing a circuit on a programmable device. The network used here is HSRA, a fat-Pyramid structure having advantages of both hierarchical structure of a tree and a mesh network [2]. The mesh network makes the shortcut connections of HSRA. More details on HSRA structure is given in section 2.

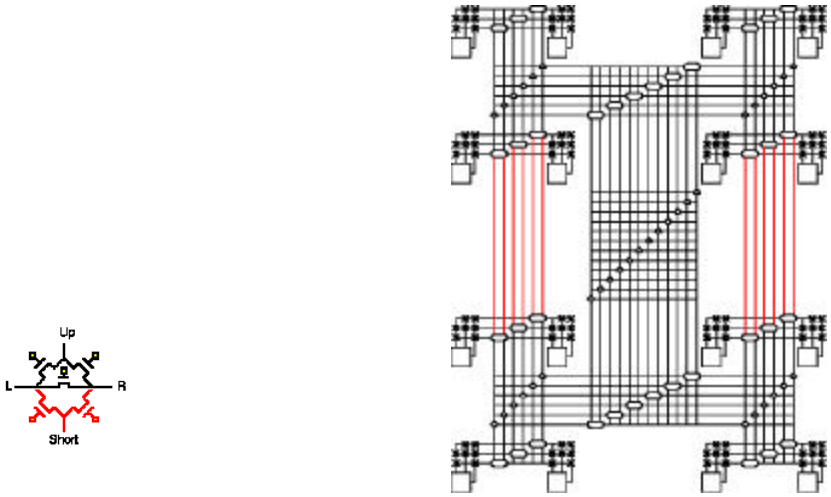
In this project I try to show the advantage of using shortcut connections in time efficiency. Shortcuts were exploited in routing in past works. Here I pulled up using shortcuts to one level upper in circuit designing, placement phase. I also show the benefit of considering shortcuts in upper levels (section 3).

Once the designed is partitioned, it is time to place it on the device. This is the part I exploit shortcuts. The point is that placing logic elements, while considering shortcuts is no more straightforward. They are not placed one after another in the order appeared in partitioning step. Now during placement I can arrange each sub-tree's children configuration so that the largest number of shortcuts is used. To do that, first I divide the design into windows, then locally try to find the best configuration of each window based on a certain cost function (section 4).

2. FIXED WIRE PROGRAMMABLE NETWORK STRUCTURE

There are numbers of fixed wire programmable network structures, like mesh structure, and different kind of tree structures, etc. The structure used in this project is HSRA, Hierarchical Synchronous Reconfigurable Array [3]. It is a fat-pyramid structure. Fat-pyramid has both fat-tree and mesh structures. It combines the advantages of both of them [2].

The logic elements are located at the base level of the tree, on the leaves. The edges of the tree are wires, and the internal nodes are switch boxes. The mesh structure adds shortcuts to this structure. Looking at switch boxes may help to understand HSRA structure better. Fig. 1a shows a switch box, located at internal node, and fig. 1b shows a 4-level HSRA structure.



(a) (b)

Fig. 1 a) A switch box structure. The wires in left and right comes from root of the children sub-trees and the wire in top goes to the upper level, the wire in the bottom comes from an adjacent sub-tree. The shortcut wire can only be connected to the left and right children and not to the upper level.
b) A 4-level HSRA structure. The black wires are tree edges and red wires are shortcuts.

3. WHY EXPLOITING SHORTCUTS

Considering shortcuts in placement step, can affect time efficiency.

Now, how can exploiting shortcuts reduce path delays?

There is especial characteristic in HSRA structure that leads to reducing path delays by using shortcut connections. It is possible that two adjacent sub-trees may not be sibling and their first common ancestor be numbers of levels upper. The common ancestor of two adjacent sub-trees may even be the root of the tree (Fig.2).

Now consider routing case, two adjacent sub-trees have connection, and only tree edges are allowed to be used. For routing, the wires go up from the source sub-tree to their common ancestor, which may be as up as root of the tree, and then com down to the sink sub-tree. Now if using shortcuts is allowed, then the path between these two sub-trees can

be routed considerably shorter, only using a single shortcut wire, not through the long wires and switch elements. In this way using shortcut leads to smaller path delays.

Now, imagine those two sub-trees were not adjacent, so connecting wires must be the tree edges. But there are cases that little change in placement can bring those two sub-trees adjacent, and again exploiting shortcuts reduces the path delays.

4. ALGORITHM OVERVIEW

4.1 Windowing

HSRA structure is highly connected by mesh and tree edges; so considering all the possible configurations of the luts at the base level of the tree is too complicated. Besides, for large designs, the number of luts is really huge, and working with this number of luts and finding the best configuration of them, would have time complexity about O(N!)! So in this case I forget about the best answer to the problem and use a greedy local algorithm.

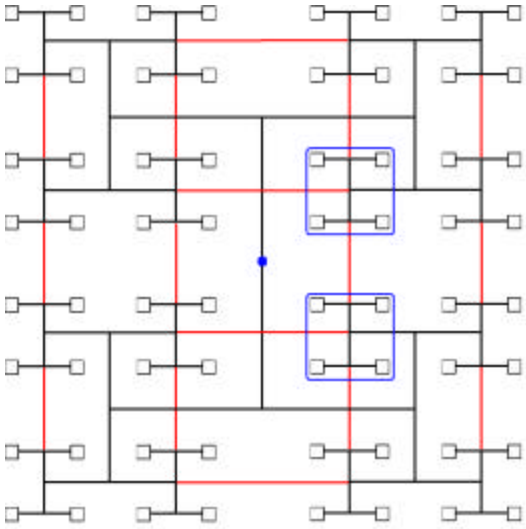


Fig.2 The two marked sub-trees, are adjacent, have shortcut connection, but their first common ancestor is the root of the tree.

First instead of solving the problem at the bottom level of the tree, I look at the upper levels of the tree, and even in each upper level, I try to focus on a small part and then go through the rest of the part in that level. Each of this smaller part is called a window. Each window is a sub-tree of the partitioned design.

For example a partitioned design would make a 7-level HSRA structure. This design can be divided into 2 windows, 1-level lower sub-trees. Each window is divided to four 3-level lower sub-trees called sub-windows [1].

The core of the algorithm is finding the best configuration of sub-windows in each window. The best configuration is evaluated by a cost function that will be defined later in this section. This process is called relaxing a window. It is shown in more details later.

Now consider the case, the algorithm is on level L of the tree. It starts relaxing the windows; it picks up each window, relaxes it and moves to the next window [1]. Once all of the windows are relaxed, the algorithm moves to the first window and repeats this process until the cost function remains unchanged within a certain range.

After relaxing the level L completely, the algorithm moves to two level lower, L-2, and partition each window to 4 new windows, or better say the sub-windows of upper level is now the windows of this level, and goes through the whole relaxing process the same way.

4.2 Priority of the tree edges and shortcuts

Shortcuts that connect sub-trees are located in different levels, that means two sub-tree are highly connected by shortcuts from different levels of the tree. Finding a configuration that uses all of these shortcuts efficiently is not possible, so I give priority to the shortcuts. This priority comes from the priority I set to the edges of tree because each shortcut is used to bypass a tree edge. The algorithm tries to use shortcuts that bypass a highest priority edge available.

Priority of the tree edges are defined based on the level of each edge, the higher level an edge is located the higher priority it gains. The motivation for this definition is that the higher level edges are the common ancestor of a larger number of luts, which means that they would be used by larger number of luts, and also are longer in length than the lower level edges [3].

When the windows are the sub-trees at level L, the most important tree edges are those that connect the sub-trees at level L and L+1, i.e. those edges that connects the windows to each other. Consequently, shortcuts that bypass these edges have the highest priority.

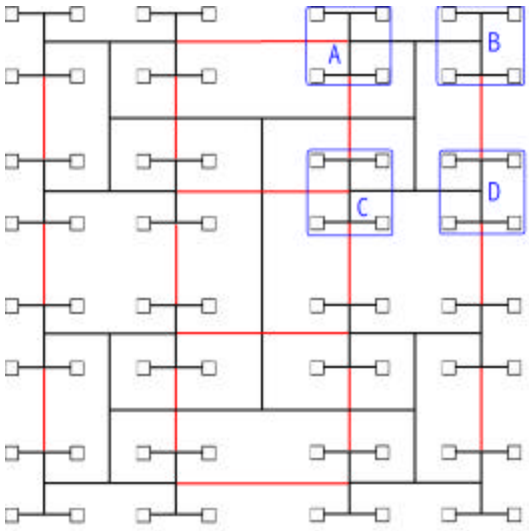


Fig.3 Sub-tree A and B are siblings, so they can exchange their places, and partitioning does not change. Sub-tree A and C belong to different sub-trees so exchanging their places, changes the partitioning.

4.3 Relaxing

Now I should introduce a method to evaluate the sub-windows configuration in each window. To evaluate the sub-window configuration, I calculate a cost function. This cost

function counts the number of potentially used shortcuts. So in this case the goal is to maximize this cost function during relaxation process.

How to find the configuration with the highest cost function?

Each window has four sub-window and they would have 24(=4!) different configurations, but not all of them keeps the partition as it was. Once we have partitioned the design to map to the HSRA, it must remain the same in the next designing steps. For example, consider the window in fig. 3, if we change the place of sub-window A with sub-window B the partitioning remains unchanged, but if we change exchange sub-window A and C then it has changed the partitioning. This means that only flipping the position of two sibling sub-trees is allowed for making a new configuration.

For making a possible configuration, we put joint in each of the internal nodes in a window. By twisting a joint, which means flipping two sub-trees a new configuration is made, and the partitioning remains the same. In fig. 4 all the 8 possible configuration of a window is shown.

How to compute the cost function?

Consider the configuration in fig. 5, for sub-window A all connections between sub-window A and sub-window 0 can be routed through shortcuts, and so can be connection between sub-window A and sub-window 1, sub-window A and sub-window 7. As you can see, sub-window A can have shortcut connections to both sub-windows of upper neighbor, but only shortcut connection to one of the sub-windows of the left neighbor. This happens because of the HSRA structure. If I define (x, y) to be the number of connections between sub-windows x and y, the cost function of the window at fig. 5 is:

Cost_function= (A,0) + (A,1) + (A,7) + (B,0) + (B,1) + (B,2) + (C,4) + (C,5) + (C,6) + (D,3) + (D,4) + (D,5).

For relaxing a window, the algorithm, calculate the cost function for all the possible configuration, and rearrange the window's sub-windows configuration to the one with highest cost function.

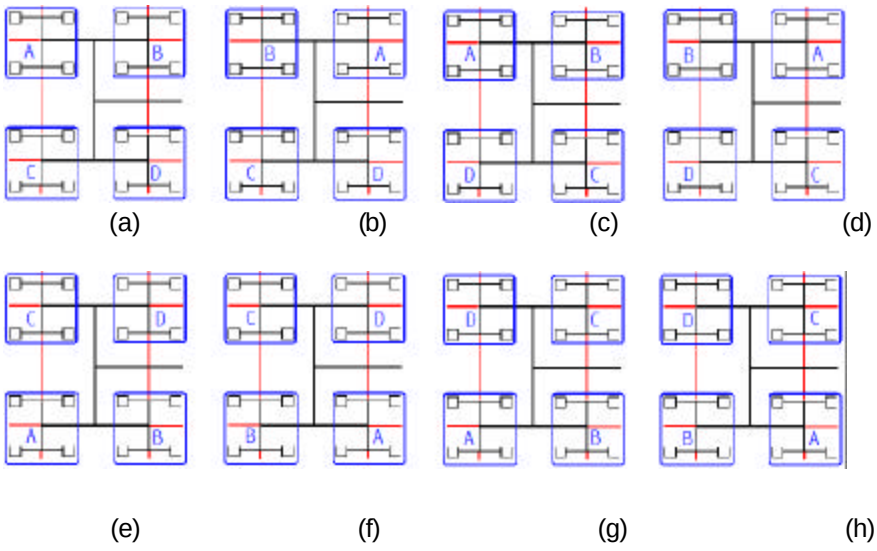


Fig. 4 Eight possible configurations for sub-windows in a window.

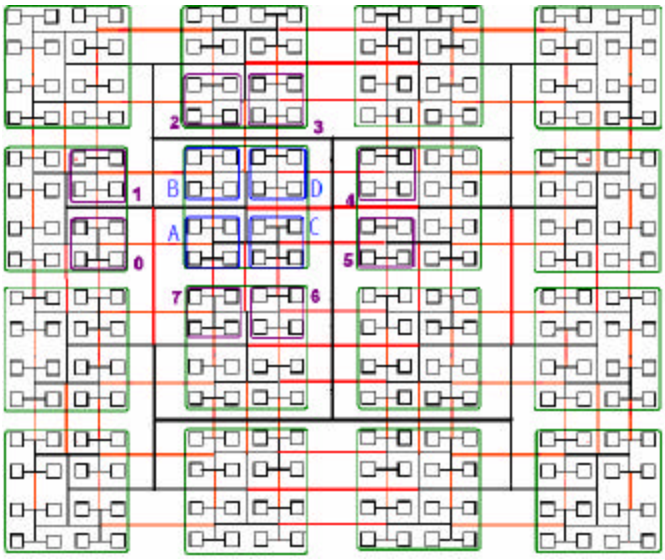


Fig. 5 Computing the cost function.

REFERENCES

[1] D.J.H. Huang, A.B. Kahng, Partitioning-based standard-cell global placement with an exact object. Proceedings of the 1997 international symposium on Physical design 1997 , Napa Valley, California, United States

[2] R.I. Greenburg. The fat-pyramid: A robust network for parallel computation. W.J. Dally, editor, Advanced Research in VLSI: Proceedings of the Sixth MIT Conference, pages, 195-213. MIT Press, 1990.

[3] William Tsu, Kip Macy, Atul Joshi, Randy Huang, Norman Walker, Tony Tung, Omid Rowhani, Varghese George, John Wawrzynek, André DeHon. HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array. Proceedings of the International Symposium on Field Programmable Gate Arrays, pages 69--78, February 1999.

How common is Turing–universal behaviour?

Ming-Shr Lin, Gaurav Saxena, Viral Shah, Geoff Wozniak

August 16, 2002

1 Introduction and Motivation

Since the introduction of Turing machines as a model of computation, many other models of computation have been introduced and have consequently been found to be equivalent to Turing machines. While most computational models are artificial constructs, mostly for mathematical consideration, some are motivated by and modeled with natural systems.

The wide variety of computational models and their equivalence to Turing machines suggests that Turing–universal behaviour may be common to most systems, notably natural ones. Intuitively, we can guess that adaptable organisms resemble Turing–universal behaviour in that given proper input, they can perform a wide variety of tasks. In fact, it could be argued that this form of "Turing–universal behaviour" is an important factor to the survival of the organism. While this intuition is highly informal, it does lend itself to further exploration of the topic.

The goal of this report is to add an element of formalism to this argument. However, it must be made clear that it is not an argument that *shows* that Turing–universal behaviour is in fact common to many systems. Rather, it is an argument that *supports* the idea that Turing–universal behaviour may be common in some systems.

Our analysis will be based upon generating random Turing machine and looking for structure within the randomly generated machine (i.e., a directed, labeled graph) that is similar to that of a universal Turing machine. In general, this is very difficult to do, since we would be looking for behaviour, not structure. We will restrict ourselves to structure only and restrict ourselves even more by looking for a specific universal Turing machine instead of any universal Turing machine (UTM). The specific Turing machine in question is Minsky's (4, 7) machine from [1]. We will denote Minsky's UTM as U_M (see Figure 1).

2 Simple Counting Argument

Our first approach consisted of the simplest approach: assume the uniform distribution over all graphs on n nodes and count the number that contain

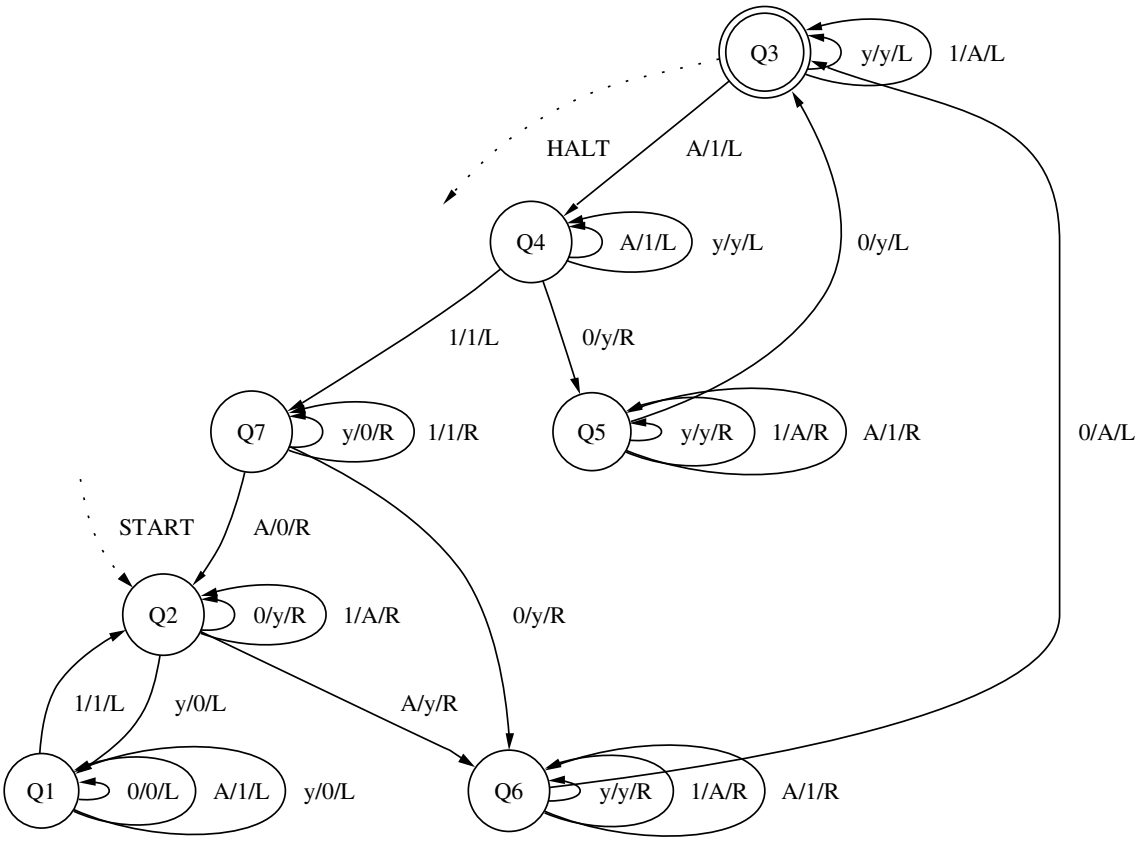


Figure 1: Minsky's (4, 7) universal Turing machine.

U_M. The graphs we considered were directed graphs that allowed up to three self–loops.

If we look at a graph as an adjacency matrix, there are a total of n^2 entries in the matrix. Not counting the diagonal, there are $n^2 - n$ entries, each of which may be set to either 0 (edge exists) or 1 (edge does not exist). Along the diagonal, we must allow up to 3 self–loops, meaning there are four possible entries: 0, 1, 2, 3. Thus, there are

$$2^{n^2-n} \cdot 4^n = 2^{n^2+n}$$

possible directed graphs on n nodes that must be considered.

Fix a numbering of the nodes in G and set U_M to be a connected component of G. Thus, there are $n-7$ nodes remaining. If we take the ratio of graphs on $n-7$ nodes with a fixed U_M as a connected component to the number of graphs on n nodes, we get

$$\frac{2^{(n-7)^2+(n-7)}}{2^{n^2+n}} = \frac{1}{2^{14n-42}}.$$

This ratio (i.e., probability) quickly approaches 0 as n goes to infinity. Even if we consider the $\binom{n}{7}$ ways of choosing 7 nodes and the 7! ways of arranging those nodes, it only results in a ratio of

$$\frac{7!\binom{n}{7}}{2^{14n-42}} = \frac{n(n-1)\ldots(n-6)}{2^{14n-42}}$$

which still results in a probability approaching 0 as $n \to \infty$.

These counting arguments grossly underestimate the correct number of graphs on n vertices that would contain U_M as a subgraph. This is due to the fact that we disallow any edges to come into the U_M component. However, this would not change the fact that the denominator is exponential with respect to n and the probability continues to tend to zero.

3 Forcing Locality

It would seem from the preceding section that Turing universality is not a very common trait. However, this was only in systems where each possible configuration (i.e., graph) is equally likely. If one observes natural systems for a short period of time, in all likelihood, it would be noticed that not every configuration is equally likely. So it would seem prudent to consider a system that favours some configurations over others.

To emulate this, we can change the distribution of probabilities on edges out of nodes. In the previous section, each edge appeared with probability 1/2. If that were narrowed to force each node to have out-degree 4 and labeled each edge, then we could weight the edges and assign probabilities based on these weights.

By looking at the Minsky machine, we see that there are many loops and the graph itself is small. Assuming we can find a weighting that favours edges going to nodes that are close to the source, it seems likely that the U_M structure will appear as the graph grows larger.

3.1 Probability of U_M in G

One aspect to locality is that as the system grows larger, it tends to make little difference as to the local connections and associations. What we are looking for is a probability distribution that favours locality and changes little as the total number of nodes grows. That is, regardless of how many nodes there are, a node x is just as likely to have an edge to a node close to x when the graph has a thousand nodes as it is when the graph has a million nodes. We would like to characterize distributions of this type.

In general, if we are looking for a specific subgraph of G, we can divide G into blocks of size l, where l is the number of nodes in the subgraph. If we assume $n = kl$, then we have k blocks of size l. Let B_i be a block, with $0 \leq i \leq k-1$ and let p_i be the probability that B_i is isomorphic to some fixed graph on l nodes. Then the probability of U_M (the fixed subgraph) being found in G is

$$\begin{aligned} Pr(U_M \subseteq G) &\geq Pr(\exists B_i \equiv U_M) \\ &= 1 - Pr(\neg \exists B_i \equiv U_M) \\ &= 1 - (1 - p_i)^k. \end{aligned} \tag{1}$$

Note that as $k \to \infty$ (assuming $p_i > 0$), then the probability of $U_M \subseteq G$ goes to 1. However, it must be the case that p_i does not depend on n. This is the crucial factor in using this argument and finding a usable probability distribution.

Designate an edge labeled by α between nodes x and y as (x, y_α). Let

$$D = e^{-|x-y_\alpha|/\lambda}$$

be a weighted distribution with parameter λ. We can see that D emphasizes locality since the size of the exponent is based on the difference (distance) between two nodes and not on the number of nodes in G and that weightings between two different edges are independent of each other. The distance between two nodes is based on their numbering (i.e., node x is considered to be a distance of r away from node $x + r$).

Since the overall goal is to find the probability of U_M in a random graph, we must normalize this weighted distribution to make it a probability distribution. That is, for some $x \in G$ and α

$$\sum_{y \in G} Pr((x, y_\alpha)) = 1.$$

To make a probability distribution for all possible edges from some node x labeled with α, we simply take the specific edge weighting over the sum of all possible weightings.

$$Pr((x, y_\alpha)) = \frac{e^{-|x-y_\alpha|/\lambda}}{\sum_{y' \in G} e^{-|x-y'_\alpha|/\lambda}} \tag{2}$$

Let a *configuration for some node* x be the set $C_x = \{(x, y_\alpha) \mid (x, y_\alpha) \in G\}$, that is, a configuration for node x is the set of all edges rooted at x. Since the probability of edges with different labels emanating from the same node are independent of each other, we have

$$Pr(C_x) = \prod_{(x,y_\alpha) \in C_x} Pr((x, y_\alpha)).$$

Let $\hat{p}_i$ be the probability of B_i being congruent to a specific isomorphism of U_M. We have

$$\hat{p}_i = \prod_{x \in B_i} Pr(C_x).$$

To get the value for p_i, sum the probabilities of $\hat{p}_i$ over all possible isomorphisms of B_i to U_M.

It is important to note that these calculations are entirely reliant upon the fact that p_i is not dependent on the number of nodes in G. For this to be true, it must be the case that the sum in (2) tends to a constant greater than zero. For example, we can consider another distribution

$$D' = \frac{1}{(x - y_\alpha)^2}$$

where x and y_α are the same as in D. As it turns out, this distribution is also favourable[1] since it converges to a constant greater than 0 and does not rely on the size of G. The reason that distributions should not rely on the size of G is that if it were, the locality argument breaks down since the probability of an edge will decrease as the size of G increases. When the sum converges, the probability of an edge tends to a constant greater than 0 as well.

3.2 Some Calculations

Empirically, we have shown that D works as a distribution.[2] We have shown that

$$\hat{p}_i \approx 1.1444 \times 10^{-19} \text{ for all } n \geq 20$$

[1] It turns out that G needs more nodes using D' as a distribution than D to have U_M appear with probability very close to 1.

[2] Matlab code for performing these calculations can be found in Appendix A.

where n is the number of nodes in the random graph G and setting $\lambda = 1$. Intuitively, it can be seen that the value of D is very small when x and y are very far apart and hence, the sum will converge as n is increased.

Since different isomorphisms produce different probabilities for $\hat{p}_i$, each $\hat{p}_i$ must be calculated for each isomorphism. However, it seems reasonable to estimate that

$$p_i \approx 7! \cdot \hat{p}_i \approx 5.767776e - 16$$

given that there are 7! ways of numbering the seven nodes in each B_i. In order for $Pr(\exists B_i \equiv U_M)$ to approach 1, we need a graph with approximately 10^{16} nodes. As p_i represents a lower bound, asymptotically, this is favourable.

4 Discussion

It is important to note that although the probability of U_M appearing in G tends to 1 as the number of nodes increases (assuming a local distribution), it does not say anything about whether or not U_M will actually get executed in G. That is, given G containing U_M, we do not know whether or not U_M is reachable in G on some given input (this is an undecidable problem). This demonstrates (one of) the shortcomings of our analysis. The analysis focused mainly on structure in randomly generated graphs representing Turing machines, however, very little about structure allows us to say much about behaviour.

Clearly, some fine points that could be expanded upon in the analysis have not been addressed. This was done mainly for simplicity. Most of the aspects that were not addressed would increase the probability. For example, our analysis only addresses a fixed choice for each block B_i and we did not consider all possible blocks in G. Also, we considered the nodes 1 and n to be very far apart. If we imagined the listing of a nodes to be a circle for the sake of locality, it makes a slight difference for the values of p_i for those blocks.[3]

Other aspects tend to significantly raise the complexity of the problem. We did not, for example, consider any other universal Turing machines besides Minsky's (4, 7) machine. Technically, there are an infinite number of these machines[4], however this becomes the problem of analyzing behaviour which, as we have stated, is difficult. Further analyses we feel would be interesting to pursue out of this include finding other distributions that exhibit a more "natural" behaviour, considering locality based on physical distance and allowing other UTMs to be present.

To digress slightly, it may be circumspect of us to address the question that makes up the title of this report in a more philosophical or general sense. A reaction that may have occurred in some readers upon reading the title is that of puzzlement at what exactly is meant by the question. The underlying

[3] The difference is negligible, but is mentioned here out of completeness.

[4] We can always add any number of "useless" states to any Turing machine and it will be an equivalent machine.

assumption touched on briefly in the introduction is that we are investigating random instances of systems that can perform universal computations. To draw an analogy between the artificial systems and the natural ones says that a natural system can perform universal computations. While this is true, it is true, if one will excuse the term, in an artificial sense. Natural systems can perform universal computations in the sense that human intervention has found ways to manipulate them using their inherent properties to make them capable of doing so. However, it is unknown whether or not natural systems actually perform universal computations in their regular environment or as part of their ritualistic survival. This report does make any claims, implied or otherwise, addressing this question as it is clearly outside the realm of a discussion of random graphs. It does raise some deep, important questions that should be pursued over time, such as how important universality is to an organism. Does universality afford an organism a better chance at survival? What happens when an organism does not have the universality trait? Is it even possible for an organism to not have the universality trait and be able to function? It is doubtful these questions can be answered immediately, but they could provide motivation for some very interesting research between biology and computer science.

5 Acknowledgments

The authors wish to thank Erik Winfree for laying the groundwork for the majority of this report as well as a plethora of insightful guidance and to Tom Knight for inspiration related to the analogy between natural and artificial systems.

References

[1] Minsky, M., *Computation: Finite and Infinite Machines*, Prentice Hall, 1967.

A Matlab code

A.1 MinskyUTM.dat

```
1 1 1 2
2 2 1 6
3 3 3 4
4 4 5 7
5 5 5 3
6 6 6 3
7 7 2 6
```

A.2 wexplocal.m

```
function w = wexplocal (node, N, lambda)
%WEXPLOCAL Compute the weight of a node based on other nodes/parameter.
%
%  WEXPLOCAL (NODE, N, LAMBDA) computes the weight of NODE based on it's
%  locality to N.  LAMBDA defaults to 1.0 if not given.  The formula is
%
%              e^((-|NODE - N|) / LAMBDA)
%
%  where NODE is a scalar and N may be a scalar or a matrix.

if nargin == 2
   lambda = 1.0;
end

w = exp (- (abs (node - N) ./ lambda));
```

A.3 sumweights.m

```
function s = sumweights (node, N, wfunc, lambda)
%SUMWEIGHTS  Sum the weights over a given number of elements.
%
%  SUMWEIGHTS (NODE, N, WFUNC, LAMBDA) sums the weights as given by WFUNC
%  over N elements as they relate to NODE.  WFUNC is called as such:
%
%    WFUNC (NODE, N, LAMBDA)
%
%  If LAMBDA is not given, it defaults to 1.0.
%
%  Note that the second argument to WFUNC (that is, N) must be able to
%  handle matrices.

if nargin == 3
   lambda = 1.0;
end

s = sum (feval (wfunc, node, N, lambda));
```

A.4 probconfig.m

```
function p = probconfig (node, ADJV, n, wfunc, lambda)
%PROBCONFIG  Compute the probability of a configuration for a node.
%
%  PROBCONFIG (NODE, ADJV, N, WFUNC, LAMBDA) computes the probability
%  of NODE having the configuration given by the adjacency vector ADJV
%  in a graph of N nodes with weighted distribution given by WFUNC and
%  adjustment parameter LAMBDA.  If LAMBDA is not given, it defaults
%  to 1.0.
%
%  See 'sumweights' for an explanation of WFUNC.
%
%  Note that N should be a scalar!  Here is an example of how to call
%  PROBCONFIG with an explanation:
%
%      probconfig (5, [5 6 6 7], 100, @wexplocal, 0.5)
%
%  This computes the probability that node 5 has four outgoing edges.
%  Specifically, it has a self loop (node 5 to node 5), two edges to
%  node 6 and an edge to node 7.  The full graph has 100 nodes and the
%  function 'wexplocal' is used to calculate the weighting of the
%  edges, with lambda parameter set to 0.5.

if nargin == 4
   lambda = 1.0;
end

denom = (sumweights (node, 1:n, wfunc, lambda))^size (ADJV, 2);
p = prod (feval (wfunc, node, ADJV, lambda)) / denom;
```

A.5 probblock.m

```
function p = probblock (block, ADJ, n, wfunc, lambda)
%PROBBLOCK Compute the probabilities of node configurations in a block.
%
%  PROBBLOCK (BLOCK, ADJ, N, WFUNC, LAMBDA) is essentially the same as
%  'probconfig' except that it computes the probabilities for entire
%  blocks.  BLOCK should be greater than or equal to 0.  ADJ should be
%  an entire adjacency matrix for the block.  N, WFUNC and LAMBDA are
%  the same as 'probconfig'.
%
%  An example is probably the best way to see how to use it.
%
%    probblock (3, MinskyUTM, 100, @wexplocal)
%
%  This computes the probability of the 4th block (recall block
%  numbering starts at zero) of a 100 node graph containing a
%  structure exactly as represented by the adjacency matrix
%  MinskyUTM.  LAMBDA is assumed to be 1.0, since it was not given as
%  an argument.
%
%  The size of a block is calculated by the number of rows in the
%  adjacency matrix.  Thus, if you pass in a 7x4 matrix, a block is
%  assumed to be 7 nodes and each node has 4 outgoing edges.  The
%  offset is calculated as (BLOCK * number of rows in ADJ).
%  Currently, the function does not check to ensure that N is
%  sufficiently large (e.g., if you want the 10th block of a 7x4
%  adjacency matrix in a graph with 30 nodes, it will return strange
%  results, since 10 * 7 > 30).
%
%  Also note that the adjacency matrix does not need to be adjusted
%  for working with a particular block.  The offset is merely added to
%  each element in ADJ to mimic the node behaviour.
%
%  PROBBLOCK returns a row vector containing the probability for each
%  node in the block.  Suppose a call to PROBBLOCK returns
%
%    [ 0.50  0.30  0.20  0.05  0.02  0.02  0.01 ]
%
%  when called on the 5th block.  This means node 7*5 + 1 = 36 has
%  probability 0.50 of occuring with the given configuration, node 37
%  has probability 0.30 and so on.

if nargin == 4
   lambda = 1.0;
end
```

```
count = 1;
probs = [];
offset = block * size (ADJ, 1);

while count <= size (ADJ, 1)
      probs(count) = probconfig ((count + offset), ...
                                 (ADJ(count, :) + offset), n, wfunc, lambda);
      count = count + 1;
end

p = probs;
```

Yelizaveta Gorstko
CBSSS, Caltech
Summer, 2002

Learning with Noise: Cellular Automata approach.

In this paper we give a brief overview of Cellular Automata (CA's) and their use in computation. We discuss potential applications of stochastic CA's and possible changes in the operators used in Genetic Algorithms that could make evolving stochastic CA's more efficient.

Yelizaveta Gorstko
CBSSS, Caltech
Summer, 2002

Introduction: Biology, Computation, and Hardware

Today attention of many scientists turns more and more often away from the sequential Von Neumann architecture towards "nonstandard" ways of computing. The main reason and motivation for this lies in the fact that too many systems around perform computations in a quite disparate way. For example, the brain, human society, and insect colonies can perform complex tasks with no "central controller" to ensure that each individual component does what it is "supposed" to do. An increasingly interesting question is to find out how these completely decentralized systems, which components act in parallel, accomplish a common goal. Inherent parallelism and highly local interactions among cells make a Cellular Automaton (CA) a natural object for attempts to carry out computation in a way similar to the one that takes place in the brain.

This paper looks at a CA as a medium for computation and asks how it could be possible to obtain a CA capable of solving a particular problem. Since it is unrealistic to manually describe the correct behavior of such a CA, we do it via a Genetic Algorithm (GA). There exists quite a big number of works devoted to evolving CA's with a GA; however, all of them consider a software implementation of the GA. A different direction is hardware implementation of a GA. The step from software to hardware requires more work than one might initially suspect. Apart from building physical circuits and gates, engineers have to deal with a big real world problem: unreliability of physical components. In the software-GA, if a cell reports that it is in a particular state, for example '0,' then it is definitely in that state, but hardware, especially with a tendency towards diminishing sizes, can have components that are less reliable. As a result a cell that has state '0' might be regarded as having the state '1' by some of its neighbors. Evolving CA's in such "noisy" conditions should differ from evolving them in deterministic environment. Our approach to modeling evolution of a CA in the presence of noise is to use a stochastic CA. Nondeterminism in the behavior allows for new definitions of genetic operators used in the GA. These operators could potentially make the algorithm more effective than if it simply disregarded the noise altogether and proceeded as if in an ideal environment. Here we will discuss several possible changes to genetic operators.

Another interesting question deals with a comparison of evolution of deterministic versus stochastic CA's. Of course, the search space for the CA with nondeterministic behavior is larger than that of deterministic one; on the other hand, we are not trying to find particular probabilities, in a sense the final goal is still a deterministic CA, but if genetic operators that manipulate probabilities instead of definite states appear more effective than the usual ones, they may yield better and faster results.

This paper is devoted to a discussion of questions and possible approaches; deciding which ones can produce good results requires practical implementations. The author of this paper implemented a short program (in Lisp) to test the newly defined genetic operators (discussed in the section on GA's). Unfortunately, one step of a GA involves multiple (about 300) runs of CA on different instances of the problem (density task in our case), and each run of CA on a regular desktop computer takes (in our implementation) approximately 45 seconds. This number implies that one run of a GA takes on the order of two weeks, and considering that fine-tuning the GA requires finding good values for a number of parameters used by the genetic operators, the realistic time

for coming up with a good algorithm is at least several months. However, this does not impede a discussion of possible ideas for future work, which could, for example, avoid the time bottleneck by using the Cellular Automata Machine (CAM) designed by Prof. Margolus and his team in the 1980s in order to speed up running of CA's by creating special architecture that used parallelism and local and uniform interconnections.

Here we will shortly discuss CA's in general and stochastic CA's in particular, then mention the main ideas behind a GA and discuss our ideas about possible changes in the genetic operators that could yield better results in a noisy environment, as well as possibly in a regular environment.

Cellular Automata

An n-dimensional Cellular Automaton consists of an n-dimensional array of cells and a finite set of states. Each cell carries one state (which in some cases may be a word that also carries information about the past state) and knows the states of several cells in its r-neighborhood, for example with r=3, each cell knows about itself, three cells to its right, and three cells to its left. In one time step all cells simultaneously update their states depending on the states of the cells in their r-neighborhood (Note: there are other ways of updating, for example "Margolus neighborhood" partitions a CA into disjoint blocks, and these blocks, not cells, are updated simultaneously; partitions alter in time, so that new blocks overlap with the old ones. This definition, useful for constructing simulations of reversible systems, is not needed in our case.).
Although, in this paper we look at CA's as a medium for computation, they are often used for simulations of physical systems. Many regard CA's as discrete counterparts to differential equations: they allow description of continuous systems with a finite number of cells that have a finite number of states and follow predefined rules. The widespread interest in CA's first arose due to their ability to simulate complex dynamics by simple local interactions.

Once it has been shown that CA's are capable of Universal computation, which means that potentially one could find a CA that was equivalent to any finite algorithm, their computational ability started to attract interest as well. CA's are usually distinguished by their behavior: some are too simple to be capable of attacking any interesting task, some are too unpredictable, for example chaotic: little differences in the initial distribution of states produces large differences in the limiting behavior, but there are some that have a potential to form interactions necessary for computation. In particular many of the later works deal with nonuniform CA's, where rules differ for cells in different positions; the possibility to have different behaviors in different parts of a CA seems to allow more freedom in defining more interesting behaviors.

Above we have described a deterministic CA, i.e. once a cell knows the neighborhood configuration (the set that consists of the states of cells in the cell's r-neighborhood), it can take one and only one state. On the other hand, a stochastic CA has cells that follow some probability distribution during the update: once a cell receives the information about the neighborhood configuration it finds this configuration in its rule table, makes a random choice between 0 and 1 and follows the corresponding probability distribution to decide which state to take based on this distribution and the random choice. For example, in a two-state CA ('0' and '1') a configuration (0 1 0 0 1 0 0) with

a corresponding probability distribution (‘0’:0.3 ‘1’:0.7) a random choice of 0.4 would make the cell change its state from 0 to 1. Stochastic CA’s have been used for various problems, such as simulations of stochastic processes, behavior of imperfect (“noisy”) systems, and prediction of a long-term behavior of deterministic systems.

In our project we did not deal with imposing “noise” on top of a deterministic CA and looking if it could be possible to still achieve expected behavior. It was noticed that CA’s that are capable of computation are incredibly “brittle,” i.e. introducing noise breaks completely their ability to perform computation. That’s why, from the very beginning we wanted to look not at the final, ready-to-use, CA, but at the means of obtaining a stochastic CA. Once such a CA is obtained, a detailed analysis of its rule table can provide information about what level of noise it can stand. The early works that deal with stochastic CA’s looked for the optimal distribution of probabilities; however, their goal was not to find a CA that is maximally robust in the presence of noise. An algorithm that introduces a penalty for rules that involve extremes, for example that have the probability of changing to ‘0’ or ‘1’ go too near to 1, can direct the CA towards being more nondeterministic. Once we have the final CA, we can look at the rule tables for each cell and analyze how well this CA can perform given the level of our background noise. The final CA can be regarded as a deterministic CA (the state corresponding to the biggest probability in the rule gives the deterministic state: this is equivalent to “majority rules” approach of running a nondeterministic CA many times, or running multiple copies of it) and the variation (probabilities of choosing other states) give the maximum interference, i.e. background noise, that could be imposed on the CA. This analysis can show several things: first, whether or not this CA would be functional at all in the imposed conditions; second, if reducing noise is costly, in which parts we can allow having more noise, as well as which parts of the CA are the most sensitive to noise and have to be protected. These possible uses of a stochastic CA serve as a strong enough motivation for looking at the ways to obtain one capable of a particular computation task.

Genetic Algorithms and Cellular Automata

GA’s get the solution (or a close approximation of it) to the problem not explicitly, by deriving it, but implicitly, through an evolutionary process. In a GA the well-known principle of the “survival of the fittest” applies not to live organisms, but to solutions of the problem. GA consists of the following main components: a fitness function that, based on the results of an individual’s performance subject to one or several tests, evaluates how close each individual comes to satisfying the requirements that the true solution should satisfy, a selection mechanism that picks out individuals for reproduction based on their fitness, and genetic operators of crossover and mutation that have to shift the properties of offspring individuals closer to satisfying these predefined requirements. In our case solution is the CA. However, dealing with multiple CA’s requires too much time. If we want to apply a GA to find this solution, we should consider rules in each cell as individuals. A natural choice for a “gene” is the probability distribution that corresponds to a particular neighborhood configuration; a “chromosome” should contain distributions corresponding to all of the configurations, i.e. the whole rule. Thus one individual here is encoded with one chromosome. We start with a population of

randomly generated rules and evolve them for about a hundred generations. The above description already pinpoints the main advantage and disadvantage of a GA. We do not need to explicitly know the structure of the solution to be able to obtain it. This makes GA's a good choice for the problems where it is hard to figure out the precise relations and dependencies, but easy to see whether or not a potential solution satisfies our requirements. On the other hand, it is often hard to derive the fitness of an individual based on its genotype, which means that it is necessary to subject it to tests and evaluate fitness based on its phenotype. More tests given to each individual make the fitness estimate better. However, running these tests may take a long time. Indeed, tests used in fitness evaluation take the biggest part of the entire GA, and this part tends to become a serious bottleneck in the case of many complex problems. A potential solution is a hardware implementation of a GA, which allows time reduction by several orders of magnitude. And, as we have mentioned before, knowledge about evolving stochastic CA's will most probably be useful for hardware implementation.

Detailed descriptions of a GA used to evolve a deterministic CA designed for a density task are given in the works by Mitchell and Sipper. Similar to their works, we have a one-dimensional CA with 149 cells designed to perform the density task: decide whether a given list of 149 0's and 1's has more '0' or not. Length of the neighborhood, r, is 1, i.e. each cell considers a list of three elements: states of its immediate left and right neighbors, and its own, as a configuration that allows updating the current state. The given problem provides initial states for the cells, and these, i.e. '0' and '1,' are all the states that we use.

A side note: here the number for the length of the problem equals the number of cells. Biological systems often amplify the signal before producing the reaction. The existing models all show increasingly poor behavior for problems where the number of 0's approaches the number of 1's. This range of poor behavior gives a threshold on sensitivity to differences in the quantity of 0's and 1's for a CA that does not amplify the input signal (problem). If a CA replicates the problem several (k) times, the difference gets multiplied by k also, thus decreasing the range of poor performance. One way that CA's could achieve better performance might be by amplifying signals and increasing the length. Of course, if more cells make it more costly, it will have to balance performance with length. This is again an optimization problem, which suggests another interesting application of a GA: to find k that gives the preset error range given the penalty for the wrong behavior and a penalty for extra cells.

In order to evaluate fitness, rules have to be subjected to tests. Here is one of the interesting characteristics of CA's: it is impossible to assess each individual rule completely on its own. The whole idea is that individual components <u>together</u> lead to some meaningful behavior, and we don't know what each individual component is supposed to be doing. Behavior of each individual is too closely interwoven with behavior of the cells in its r-neighborhood. The only way to assign fitness to a rule is to run the entire CA on "many" problems, each of which has a known solution, and see if the given cell reaches the correct "limiting" state in each case. The ratio of the number of correct final states to the total number of tests gives a good approximation of the fitness of a cell. Of course, since the simulation has to end in a reasonable time both "many" and "limiting" have to be given finite values that are both sufficient and realistic. The numbers used here are 300 problems for "many" and 320 steps for "limiting."

The problems are randomly generated. The number of problems in the "less 0's" and "less 1's" categories are also picked out randomly from a uniform distribution. This ensures that a cell that simply always sticks to one of the choices, i.e. exhibits the "1/2 behavior," does not appear as a good one to the genetic algorithm. It is highly improbable that a CA with performance of 1 (a 100 per cent correct) exists; it definitely does not among uniform, one dimensional CA's. The rule derived manually by Gacs, Kurdyumov, and Levin in 1978 for a uniform CA with r equal to three, defined by the "majority vote" among: if current state is '0': you, left neighbor, cell 3 sites to the left, and if current state is '1': you, right neighbor, cell 3 sites to the right, performs as well as 0.98, and the CA's obtained via GA's get as high as 0.95-0.96. This shows that expected behavior of the stochastic solution should definitely be much higher than 0.5 and should even be approaching 0.9's.

The selection mechanism is responsible for both pushing the evolving population in the desired direction and making sure that it does not exhibit the "fast convergence" phenomenon, which happens when the population converges too quickly to an individual that dominates the rest of population, but does not satisfy the requirements well enough. The dominance of its genes may delay or prevent better solutions from appearing during one run of a GA, i.e. in a hundred generations. In the works by Mitchell, fast convergence is mentioned as one of the biggest problems. They always choose the best individuals for reproduction; however, this selection impedes transitions to new "stages" (the current stage changes with evolvement of a radically new behavior). For example, CA that has discovered the one-half behavior, under a fitness function that does not punish this behavior well enough, may have a few cells that have this behavior, and if the other cells perform worse, the selection mechanism may quickly force all the cells to be their copies, thus eliminating diversity in the population and taking away the whole meaning of a genetic algorithm, which is not designed to obtain all the diversity in the gene pool exclusively from the effect of genetic operators, in particular mutation. In order to avoid this problem, we have implemented the "roulette wheel" selection, where the probability of being picked out are in direct proportion with fitness, and yet the best individuals are not guaranteed a chance to contribute to the creation of the next generation.

Finally, the genetic operators, which receive the set of parents from the selection function and output the set of offspring, were the main point of interest of our project. In all of the works mutation played a tiny role and all the emphasis was given to the crossover operator. The probability of mutation was a small constant, independent of the fitness of an individual. We decided to explore the role of mutation in more detail. Sipper insists on introducing locality from the very first stages: the parent cells for an offspring come only from its neighborhood, which is similar to what happens in real life. Initial assignment of rules is random, so on the first stages the cell finds itself evolving with a random, but heavily reduced gene-pool. Neighborhoods overlap, so eventually changes do propagate, but, maybe, for nonuniform CA's it could be more efficient also to employ mutation to solve the problem of gene reduction. After all, the species on initial stages of the development of life on the Earth were prone to mutations. Apart from individual fitness values, it might be useful to define for each cell the fitness of its neighborhood (for example, just an average of fitness values of all cells in its r-neighborhood). Mutation operator can take the neighborhood fitness and, based on it, adjust the probability of mutating (changing a random gene of) the rule. Instead of a constant line at some level,

the mutation operator with probability as a function of the neighborhood fitness, looking like a concave function (through the (0,1) point) that sharply declines to this constant line, could help balance gene diversity and reproduction locality for the CA. The role of the crossover operator, which swaps pieces of two chromosomes at the random breaking point, is to direct the population towards the higher overall fitness. An interesting problem here arises due to the ordering of the genes in the chromosome. Success of the crossover operator depends on whether or not it can preserve structures in the neighborhood relations that are important for the computation. Of course, we don't know which exactly dependencies to preserve. The standard way to arrange the genes in the chromosome (rule) is by the ordering the neighborhood configurations lexicographically, as words. Since this ordering is imposed by us and most probably is not of any significance to the CA, this choice is equivalent to picking a random order once and keeping it for the rest of the evolutionary process. It is possible that varying this order could bring better results. Instead of choosing a random point and switching the pieces of two chromosomes, we pick a random number for the total number of genes involved in the crossover, as a proportion of the total length of the rule and randomly pick out this number of genes from the rule. Varied random crossover instead of the fixed one may give better fit offspring in the long run. The next question deals with what we should do, once we have compiled these two lists of genes that are involved in the crossover operation. The standard method is simply to swap them. However, we are dealing with a stochastic CA, moreover with a possible plan to impose the noise conditions on it. It might be useful to consider whether or not the two genes (note: not the two original chromosomes) that interact during crossover have an equal say in this process. Suppose we know that one cell is definitely more reliable than the other, where reliability means an accurate report about its own state. Suppose further that it is possible to test cells before the GA starts, and we know that one cell makes a mistake with a frequency of 0.3 and another - with a frequency of 0.7. Unlike deterministic CA's, stochastic CA's allow for an arithmetic manipulation of the probabilities in each gene, which suggests the following possible crossover operation: take the probability distributions of both genes and output the weighted (by 0.7 and 0.3 - the probabilities of correct outputs) average of these probability distribution. It might also be useful to make weights depend on the neighborhood fitness as well. The weighted averages may help to take into account both the reliability of cells and their relative fitness. Biological crossover involves simple switching of genes. Yet nature created such notions as dominant and recessive genes, and these ideas are absent in algorithms that work with a single-stranded chromosome. There are some works devoted to "dominant crossover" in genetic programming; however, typical GA's do not implement any form of dominance outside of the selection mechanism. Weighted averaging may be one of the possible ways to accomplish this for stochastic CA's.

Conclusion

We have discussed the possible uses of stochastic CA's and ideas that could help efficient implementation of GA's that could evolve them. All of the suggestions are not hard for implementing in any language that handles lists well, and with an access to a

CAM machine, which could eliminate the time constraint, are quite realistic for practical experiments. These experiments would show which of the ideas are indeed beneficial and which only complicate the algorithm without making it at all more efficient. We hope that, apart from giving a short background and overview of CA's and GA's, this discussion can lead to some serious attempts of evolving and studying stochastic CA's.

References

Lee et. al.

Adaptive Stochastic Cellular Automata: Theory

Adaptive Stochastic Cellular Automata: Applications

(Physica D: Nonlinear Phenomena. Cellular Automata: Theory and Experiment, 1990)

Mitchell, M. et. al. :

Revisiting the edge of Chaos: Evolving Cellular Automata to Perform Computations

Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments

Evolving Cellular Automata with Genetic Algorithms: A review of Recent Work

Sipper, M. :

Co-evolving Non-Uniform Cellular Automata to Perform Computations

Co-evolving architectures for cellular machines

Green, Kirley :

Connectivity and Catastrophe - towards a general theory of evolution

Investigation of a Cellular Genetic Algorithm that Mimics Landscape Ecology

General:

Gacs*, Reliable Cellular Automata with Self-organization*

J. von Neumann*, Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components*

Acknowledgments

This paper is a result of the ***"Computing Beyond Silicon" Summer School*** held at the *California Institute of Technology* in 2002. Without the lectures and discussions during the CBSSS the author would not have had a chance to find out and, as a result, think about computation with Cellular Automata.

The author would like to thank the organizers of the CBSSS program:
Dr. Andre DeHon and **Dr. Erik Winfree** whose support and criticism incited and directed the investigation about stochastic CA's and GA's.
And also **Dr. Margolus** whose lectures and discussions showed CA's at work, as models of physical systems, and encouraged the interest in the object that has such strictly local interconnections and yet is capable of such unison and coordination.

On the Designing of Proteins That Target Specific DNA Sequences

Computing Beyond Silicon Summer School - Caltech, July 2002
- project report -

Mirela Andronescu[1], Tiberiu Dan Onuta[2], and Yingley Zhao[1]

[1] Department of Computer Science
University of British Columbia
{androones,szhao}@cs.ubc.ca
[2] Department of Physics
University of Notre Dame
tonuta@nd.edu

Abstract. In this paper we want to find out what are the mechanisms that we could use in order to genetically engineer proteins such that they change a gene expression. It is already known how to use genetic engineering to produce proteins with a given amino acid sequence. Once we have the appropriate design of proteins, we might be able to use them in therapies for deseases based on genes' activity.

1 Introduction

The understanding of protein-DNA interaction is crucial for prediction of DNA-binding specificity of the factors which control transcription, recombination, restriction and replication. Gene expression in organisms is controlled by a variety of regulatory proteins. Structural analysis of protein-DNA complexes has revealed that the same amino acids often interact with different bases and vice versa [1]. Thus, no general rules exist yet to explain how proteins discriminate among DNA sequences or to predict their target sites. On the other hand, the amount of structural information on protein-DNA recognition has been increasing rapidly.

DNA-protein interactions occur at specific sites of the DNA, expressing or not a gene. A gene is a sequence of DNA that codes for a diffusible product. This product may be one or several proteins, as in the case of the majority of genes, or may be a type of RNA, as in the case of genes that code for tRNA and rRNA. The main characteristic is that the product obtained after transcription diffuses away from its site of synthesis to act elsewhere [2]. The first step in gene expression is always the same: the sequence along one of its strands is transcribed into a linear molecule of messenger RNA (mRNA). For our purposes, we define a gene as being *ON* if it is *expressed*, i.e. it creates protein(s). As opposed to a gene that is *ON*, we say that a gene is *OFF* if it does not produce any protein, thus having no role in the cell.

Broadly speaking, eucaryote gene expression can be achieved at any step on the pathway leading from DNA to protein [3]. The controlling of the gene expression can be done at any of these stages as follows:

1. controlling when and how often a given gene is transcribed;
2. controlling how the primary RNA transcript is spliced or processed;
3. selecting which mRNAs in the nucleus will be exported to the cytoplasm;
4. controlling which mRNAs in the cytoplasm are translated by ribosomes;
5. degrading certain mRNA molecules in the cytoplasm;
6. selectively activating or inactivating protein molecules after they have been created.

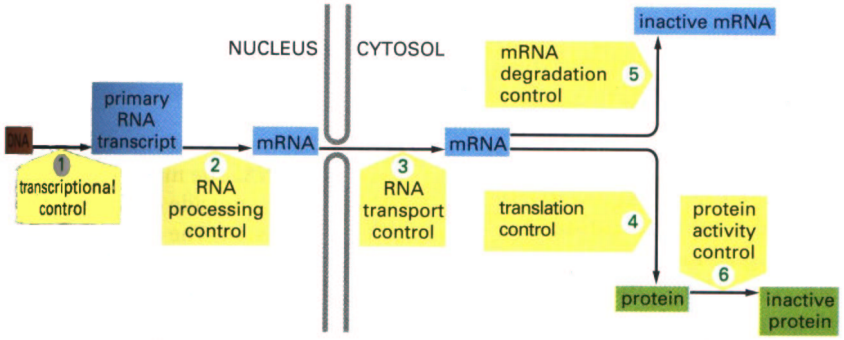


Fig. 1. The six steps at which eucaryote gene expression can be controlled.

In the process of its transcription to mRNA, the gene is helped by an enzyme called RNA polymerase. The process begins with the binding of this enzyme near the beginning of a gene to a site called *promoter*, whose length is roughly 60 bp. Each promoter points RNA polymerase in either one or the other direction along the DNA helix. While the enzyme moves to a given direction, it copies one of the strands into a new mRNA strand. The polarity of the mRNA chain is opposite to that of the DNA template strand.

RNA polymerase can be helped or hindered in its attempt to transcribe a gene by regulatory proteins that bind to sites on the DNA called *operators*. A *negative regulatory protein* prevents transcription and a *positive regulator* stimulates the transcription of a gene. A particular regulatory protein binds to a specific operator site(s) on a DNA molecule.

In this work, we want to change the role of a gene, by turning it *OFF* when it was *ON* by default, and by turning it *ON* when it was *OFF*. The **motivation** behind this work is mainly related to therapies for diseases that rely on genes' activity. We concentrate on achieving this behaviour by acting at the level of transcription initiation, the first step out of the different aforementioned stages. The most important class of transcription factors is known as zinc finger DNA-binding proteins (ZFP), that bind to the operator of a specific gene. The ability to engineer zinc fingers to specifically turn *ON* or *OFF* a gene function was first discovered by Carl Pabo at the Massachusetts Institute of Technology. Sangamo

BioSciences Inc. has also created ZFP transcription factors that can control gene expression, and consequently, cell function [5].

The remainder of this paper is organized as follows: section 2 introduces three related works that are relevant for our purposes. In section 3 we try to understand the factors that are definitory for protein-DNA interactions, focusing on zinc fingers and publicly available protein databases. We also propose some novel ideas on how to design DNA-binding proteins to turn a specific gene *ON* or *OFF*. Finally, section 4 presents conclusions and gives ideas of future work.

2 Related Work

Sarisky et al. [6]

Sarisky et al. [6] created a tool called *ORBIT* (Optimization of Rotamers by Iterative Techniques)to design proteins that target specific DNA sequence. Using the yeast transcription factor GCN4 as a model system, they first computationally generate sequences predicted to bind with high affinity to the wild type DNA target. The proteins are expressed in E. coli. and experimentally characterized by gel shift electrophoresis and DNAseI foot printing.

They elucidated some of the factors which are important for DNA binding with site-specific recognition as well as to develop a methodology for generating novel proteins with high affinity for target DNA sites. They developed a general method to computationally select protein sequences that recognize a target DNA sequence. They use a force field developed for use in the protein design process ORBIT. The force field used in ORBIT includes terms for van der Waals contacts, solvation (a benefit for burial of hydrophobic surface area, a penalty for burial of polar surface area or exposure of hydrophobic surface area), electrostatics, and hydrogen bonding.

They used a force field and a design methodology that allow generation of protein sequences to bind to any target DNA sequence.

They describe a general methodology of generation of novel proteins that bind site-specifically to DNA. This approach will allow us to elucidate the relative importance of various force to the formation of protein-DNA complexes. The ability to target specific DNA sequence with small proteins may also prove useful for therapeutic or diagnostic purpose where binding of a specific DNA sequence is necessary.

Jamieson et al [7]

In [7], Jamieson et al. use random mutagenesis and phage display to alter the DNA-binding specificity of Zif268, a transcription factor that contains three zinc finger domains. Four residues in the helix of finger 1 of Zif268 that potentially mediate DNA binding were identified from an X-ray structure of the Zif268-DNA complex. A library was constructed in which these residues were randomly mutated and the Zif268 variants were fused to a truncated version of the gene III coat protein on the surface of M13 filamentous phage particles.

They were unable to enrich for clones that bind to five other binding sites (ACG, CCG, CGC, ATA and TAT), suggesting modification of just these four

residues in finger 1 may not allow it to adapt to all DNA binding sites. The studies show that it is possible to isolate zinc fingers by Phage display that distinguishes operator sequences that differ by a single base change. Moreover, such selection methods should aid in clarifying rules for zinc finger-DNA recognition.

The paper suggests that phase display is a powerful tool for sorting libraries of zinc finger proteins for binding to different operator sequences. Specific enrichments were seen after several rounds of sorting with some operator sequences. When clones were analyzed after several rounds of selection for a give operator, they often had similar sequences and differed from clones isolated using different operator sequence. Moreover, some operator sequences gave no enrichments for binding phage, and when sequences from one of these (ACG) were analyzed, little consensus was observed.

Although the specificity advantages determined for the few clones analyzed are mild, the clones were still powerful selected. There are limitations of the phage display method that need to be emphasized.

Bulyk et al. [8]

In [8], Bulyk et al. describe a DNA microarray-based method to characterize sequence-specific DNA recognition by zinc-finger proteins. A phage display library, prepared by randomizing critical amino acid residues in the second of three fingers of the mouse Zif268 domain, provided a rich source of zinc-finger protein with variant DNA-binding specificities. Microarrays contain all possible 3-bp binding sites for the variable zinc fingers permitted the quantitation of the binding site preferences of the entire library, pools of zinc fingers corresponding to different rounds of selection from this library, as well as individual Zif268 variants that were isolated from the library by using specific DNA sequence. The results demonstrate the feasibility of using DNA microarrays for genome-wide identification of putative transcription factor-binding sites.

They use DNA microarray to examine the spectrum of binding-site specificities of a collection of Zif268 mutants selected from a phage display library of the second finger. Quantitative measurements of more than 750 DNS-protein interactions were gathered from 10 different microarray-binding assays by using wild-type Zif268, four mutants and seven pools of mutants.

3 Description

As we mentioned in the Introduction, a *negative* control of the gene is possible, as well as a *positive* one [2]:

- A *negative control* prevents a gene from being expressed. Figure 2 shows a gene that has the state *ON* by default. The transcription is initiated by RNA polymerase that recognizes the gene's promoter. If a *negative regulatory protein*, also called *repressor*, perfectly binds to the operator, RNA polymerase cannot initiate the transcription, therefore turning the gene expression *OFF*.
- RNA polymerase sometimes cannot initiate the transcription by itself. In this case, a *transcription factor* is required to assist it in binding to the promoter, having an important role in the *positive control* of the gene. The gene showed

in Figure 3 is inactive by default. Several factors are needed to bind to sites in the vicinity of the promoter, in order to enable RNA poymerase to initiate the transcription. The ultimate effect of these factors is thus to turn the gene *ON*.

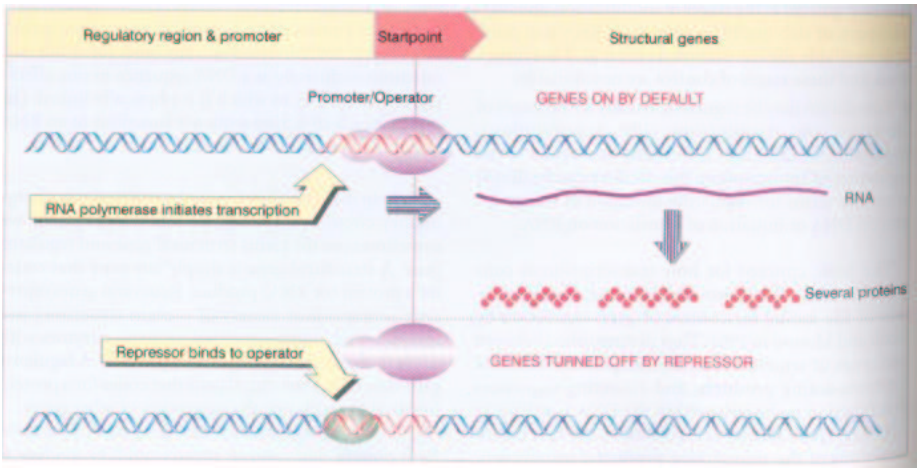


Fig. 2. In negative control, a repressor binding to the operator of the gene can turn the gene *OFF*.

Proteins fold into characteristic shapes, determined by their amino acid sequences. Different amino acid sequences usually fold into different shapes, and protein folding is a hard and yet unsolved problem. Nevertheless, small structural motifs are found in many proteins, the most common and studied one being the α helix. Figure 4 shows an α helix that fits perfectly into the major groove of a DNA strand.

3.1 Regulatory Proteins

The transcription factors have been researched and analysed. Therefore, we can compare them and find that common types of *motifs* are responsible for binding to a DNA molecule. The motifs are short, comprised a small part of the protein structure, and are also responsible for activating transcription via interactions between proteins of the transcription machinery. There exists detailed information about several groups of proteins that control transcription by using different motifs to bind to DNA molecule.

A classification of DNA-binding protein structures includes:

1. *Zinc containing DNA binding domains*
   - *Zinc fingers* are comprised of an α helix and a β strand. Zn binds two histidines in the α helix and two cysteines in the β strand. Basic amino

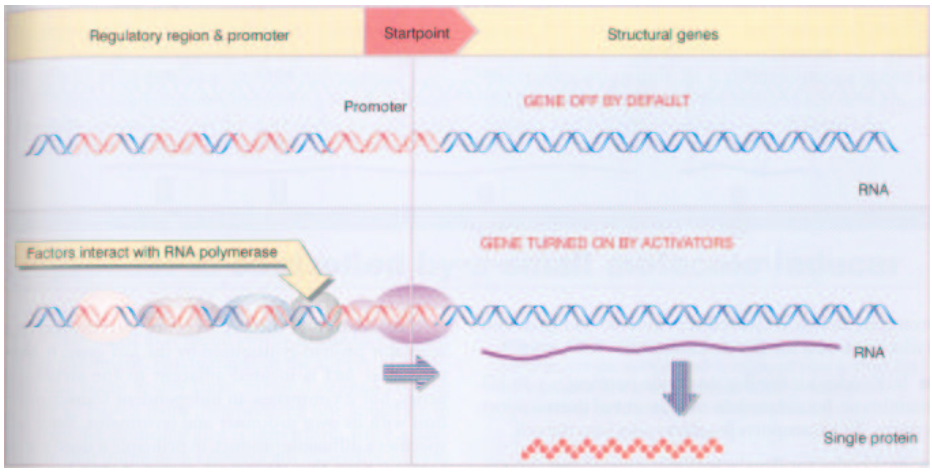


Fig. 3. In positive control, certain factors are needed to help RNA polymerase to initiate the transcription, thus turning the gene *ON*.

acids in the α helix contact nucleotides in the major groove. Each of the three fingers (see Figure 5) from Zif268 contact nucleotides and phosphates via basic residues. The β strand of Zn fingers positions the α helix in the major groove. Zn finger proteins with multiple fingers typically bind as monomers.

- *Ga14* binds a upstream activating sequences as a dimer. It uses six cysteines and two Zn ions to bind DNA. Basic residues in the α helix makes contacts with bases in the major groove. Dimerization motif is comprised of a coiled α helix.
- *Nuclear receptors* interact with hormone ligands and bind to hormone response elements to activate transcription. The nuclear receptors are always in the nucleus and bind target sequence and repress transcription

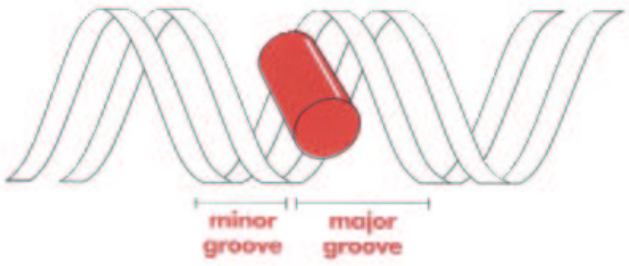


Fig. 4. Minor and major grooves of a DNA strand, and an α helix site of a protein, that fits the major groove.

in the absence of hormone, but activates transcription in the presence of hormone. Binding occurs at half sites separated by 3bp using the recognition α helix.

2. *Homeodomains* are comprised of three helices where the first two form a *helix-turn-helix* and the third is the recognition helix. They have weak binding specificity on their own.
3. *bZIP and bHLH domains* both domains combine DNA binding and dimerization. The "b" in the front of acronyms stands for the basic DNA binding. The ZIP domain, also called *leucine zippers*, is an α helix having leucines every seven amino acids, so they lie on the same side of the helix. Dimerization is mediated by a "coiled coil" formed by interactions among leucines. bZIP binds DNA like forceps. bHLH (helix-loop-helix) also forms dimers via α helix interactions which position the basic region in the major groove. There is an independence of domains, that is, the activation and DNA binding domains are modular and can be swapped to change the specificity of binding and activation or repression.

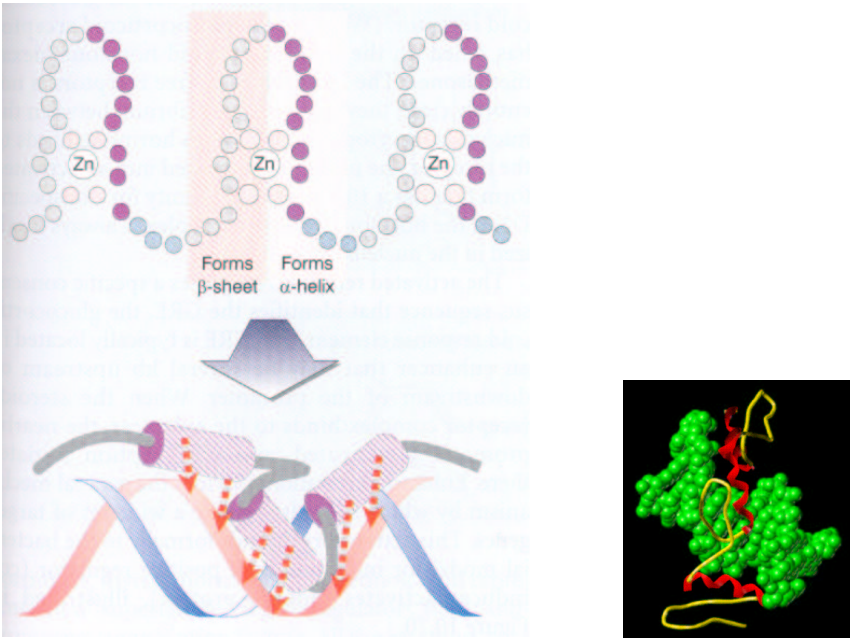


Fig. 5. Zif268 Zinc Finger-DNA complex (1AAY): a schematic view and a 3D figure

3.2 Physical Bonds between proteins and DNA

The binding between a DNA strand and a protein(s) is on hydrogen-bond bases (see Figure 6). Direct evidence for the existance of individual hydrogen bonds

in such biomolecules has been provided by the detection of hydrogen bond scalar couplings in DNA, proteins, and their complexes. These scalar couplings are electron-mediated and can be used to identify the three partners involved in the hydrogen bond, i.e. the donor, the acceptor and the proton. The size of the hydrogen bond scalar couplings correlates with the strength of the hydrogen bond and with the chemical shift of the proton involved in the H-bond. Because of the characteristics of the hydrogen bonds, the proteins have high affinity for a specific DNA sequence and a low affinity for any other DNA sequence. Actually, there is only one high-affinity site in the gene: the operator. The remainder of the gene provides low-affinity binding sites.

In the DNA-protein interactions, the hydrogen bonds appear between hydrogen atoms belonging to DNA bases (or amino acids from protein) and N or O atoms from amino acids (or DNA bases).

In addition, there are other standard force fields that should be taken into account in order to explain the specificity of DNA-protein interactions. In some models, it turns out that favorable van der Waals terms have the role of overpowering the hydrogen bonding terms. However, this is unlikely to be true, since the specific hydrogen bonds are more valuable than non-specific van der Waals contacts for specific binding.

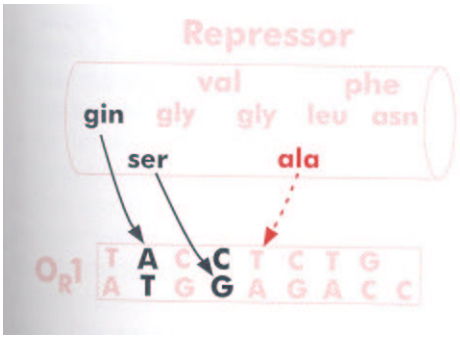


Fig. 6. An example of physical bonds between a protein and a DNA sequence

3.3 Examples of proteins that recognize specific DNA sequences

Genetic analyses have started in bacteria since 1950s, providing the first evidence of the existance of gene regulatory proteins, many such proteins being discovered. Hundreds of DNA sequences have been also identified, each recognized by a different gene regulatory protein or by a set of related gene regulatory proteins. Table 1 shows some examples of such proteins, the length of their amino acid sequence, their code in the Protein Database [9], along with the DNA sequences that they recognize [3].

Organism	Protein name	No. AA	PDB Id [9]	DNA sequence recognized
Bacteria	lac repressor	170	1L1M	AATTGTGAGCGGATAACAATT TTAACACTCGCCTATTGTTAA
Bacteria	lambda repressor	87	1lmb	TATCACCGCCAGAGGTA ATAGTGGCGGTCTCCAT
Yeast	GAL4	170	1D66	CGGAGGACTGTCCTCCG GCCTCCTGACAGGAGGC
Mammals	GATA-1	76	1GAT	TGATAG ACTATC

Table 1. Some Gene Regulatory Proteins and the DNA sequences they recognize

3.4 Data Banks

Several databases that show protein-DNA interactions currently exists, being updated continuously.

Protein Data Bank [9]

The Protein Data Bank was established at Brookhaven National Laboratory [10] in 1971 as an archive for biological macromolecular crystal structures. Since October 1998, the PDB has been managed by the three members of the Research Collaboratory for Structural Bioinformatics (RCSB) - Rutgers, The State University of New Jersey, the San Diego Supercomputer Center at the University of California, San Diego, and the National Institute of Standards and Technology.

PDB contains techniques of X-ray crystal structure determination, NMR, cryo-electron microscopy and theoretical modeling.

Nucleic Acid Database [11]

The Nucleic Acid Database (NDB) was established in 1991 as a resource to assemble and distribute structural information about nucleic acids. Over the years, the NDB has developed generalized software for processing, archiving, querying and distributing structural data for nucleic acid-containing structures.

Structures available in the NDB include RNA and DNA oligonucleotides with two or more bases either alone or complexed with ligands, natural nucleic acids such as tRNA and protein-nucleic acid complexes. The archive stores both primary (such as crystallization information, data-collection etc.) and derived information about the structures.

Protein-Nucleic Acid Complex Database [12]

This database contains structural data of protein-nucleic acid complex, classified according to recognition motif of proteins and DNA forms involved in the complex. It also helps researchers to understand the relationship between the structure, property and function of biomolecules. The structural data are taken from Protein Data Bank [12], and implemented into a relational database.

Valuable research has been done to understand the protein-nucleic acid interactions. Researchers collected experimentally observed thermodynamic data of protein-nucleic acid binding, and created a relational database called ProNIT [13], that is available on-line. It contains several important thermodynamic data for protein-nucleic acid binding, as well as structural and qunatitative binding data.

The database includes protein-nucleic acid binding data from scientific articles, and it currently contains 2514 entries. Using ProNIT, it would be possible to establish a relationship between thermodynamics and structueal changes upon the formation of protein-DNA complexes [13].

4 Conclusions and Future Work

In this work, we tried to find the mechanisms based on DNA-binding proteins that could change a gene expression, and also the specific factors that influence this machinery. Future work would imply a better understanding of the protein-DNA interactions and eventually the creation of a computer program able to design one or more proteins that would change the expression of a given gene.

Hydrogen bonds implied in protein-DNA interactions are characterized by a scalar force field. The latter can be characterized by a potential function, whose form is roughly known. As this potential field is, in our goal, an essential part for computing, we will try different mathematical forms such as: Toda potential, modified Toda potential, Morse potential and Lennard-Jones potential. As soon as we obtain some data using these theoretical models, we will compare them with the experimental results from the public databases (such as ProNIT etc.). Thus, we want to figure out which one of the aforementioned potential forms is the most appropriate for our goals. As soon as we know this, we can create a computer program to design proteins that target to the given DNA.

References

1. L. A. Mirny, M. S. Gelford, *Structural Analysis of Conserved Base Pairs in Protein-DNA Complexes*, Nucleic Acid Research, 2002, Vol. 30, No. 7, p. 1704-1711.
2. B. Lewin, *Genes VII*, Oxford University Press, 2000, p. 231-318, 649-684.
3. B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, J. D. Watson, *Molecular Biology of the Cell*, Third Edition, p. 401-417.
4. Mark Ptashne, *A Genetic Switch*, Second Edition, p. 33-48.
5. Sangamo BioSciences Inc. `http://www.sangamo.com`
6. Sarisky et al., *Computational design and experimental validation of novel DNA binding.*
7. Jamieson et al., *In Vitro Selection of Zinc Fingers with Altered DNA-Binding Specificity*, Biochemistry 1994.
8. Bulyk et al., *Exploring the DNA-binding specificites of zinc fingers with DNA microarrays.*
9. Berman et al., *Protein Data Bank*, Acta Cryst. (2002). D58, 899-907 `http://www.rcsb.org`
10. Brookhaven National Laboratory `http://www.bnl.gov/`
11. H. M. Berman, J. Westbrook, Z. Feng, L. Iype, B. Schneider and C. Zardecki, *The Nucleic Acid Database*, Acta Cryst. (2002). D58, 889-898

12. Protein-DNA Recognition Database
`http://www.rtc.riken.go.jp/jouhou/3dinsight/recognition.html`
13. Prabakaran et al., *Termodynamic Database for Protein-Nucleic Acid Interactions*, Bioinformatics, Vol. 17 no. 11 2001, p. 1027-1034.
`http://www.rtc.riken.go.jp/jouhou/pronit/pronit.html`
14. Protein Information Resource `http://pir.georgetown.edu`

Quantum Entanglement as a resource for Quantum Communication

Murali Kota
E15-430, 20 Ames Street
Massachusetts Institute Of Technology
Cambridge, MA 02139

Over the past few years radically new ways of computing and communication have been found that exploit quantum mechanical phenomena of physical systems.. Quantum entanglement is one such phenomenon that happens to be the heart of quantum computation and communication. We review some of the very surprising consequences of entanglement for quantum communication.

Man has always been discovering and inventing things to make many of his tasks in life simpler. Computation is one such task that the human race has actively pursued over the past few years to solve many problems that occur in real life. Over the past few decades there has been a phenomenal progress in the power of computational devices using silicon based integrated circuits. This extraordinary technological progress based on the laws of classical physics cannot continue indefinitely as the size of the devices would reach atomic scales soon. At these extreme regimes of natures, the functioning of these devices would be governed by quantum mechanics. Physicists and computer scientists have come up with ideas that exploit quantum mechanical behaviour of these devices to show radically new forms of computing and communication with quantum mechanical states of physical systems. In this report we review quantum entanglement as applicable to quantum communications.

I. QUANTUM ENTANGLEMENT

The simplest example of an entangled state is a two-qubit entangled state. This state can be mathematically represented as $\psi = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ where $|0\rangle$ and $|1\rangle$ correspond to spin-up and spin-down states of nuclei or horizontal and vertical polarization of photons respectively. The state ψ is entangled as the probability that the first bit is measured to be $|0\rangle$ is $1/2$ provided the second bit has not been measured. However, if the second bit had been measured, the probability that the first bit is measured as $|0\rangle$ is either 1 or 0, depending on whether the second bit was measured as $|0\rangle$ or $|1\rangle$ respectively. Thus the probable result of measuring the first bit is changed by a measurement of the second bit. Mathematically entangled states cannot be written as a tensor product of individual states.

A The EPR Paradox

Einstein, Podolsky and Rosen proposed a gedanken experiment that uses entangled particles in a manner that seemed to violate fundamental principles relativity. Imagine a source that generates two maximally entangled particles $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, called an EPR pair, and sends one each to Alice and Bob.

Alice and Bob can be arbitrarily far apart. Suppose that Alice measures her particle and observes state $|0\rangle$. This means that the combined state will now be $|00\rangle$ and if now Bob measures his particle

he will also observe $|0\rangle$. Similarly, if Alice measures $|1\rangle$, so will Bob. Note that the change of the combined quantum state occurs instantaneously even though the two particles may be arbitrarily far apart. It appears that this would enable Alice and Bob to communicate faster than the speed of light. Further analysis, as we shall see, shows that even though there is a coupling between the two particles, there is no way for Alice or Bob to use this mechanism to communicate.

There are two standard ways that people use to describe entangled states and their measurement. Both have their positive aspects, but both are incorrect and can lead to misunderstandings. Let us examine both in turn.

Einstein, Podolsky and Rosen proposed that each particle has some internal state that completely determines what the result of any given measurement will be. This state is, for the moment, hidden from us, and therefore the best we can currently do is to give probabilistic predictions. Such a theory is known as a local hidden variable theory. The simplest hidden variable theory for an EPR pair is that the particles are either both in state $|0\rangle$ or both in state $|1\rangle$, we just don't happen to know which. In such a theory no communication between possibly distant particles is necessary to explain the correlated measurements. However, this point of view cannot explain the results of measurements with respect to a different basis. In fact, Bell showed that any local hidden variable theory predicts that certain measurements will satisfy an inequality, known as Bell's inequality. However, the result of actual experiments performing these measurements show that Bell's inequality is violated. Thus quantum mechanics cannot be explained by any local hidden variable theory.

The second standard description is in terms of cause and effect. For example, we said earlier that a measurement performed by Alice affects a measurement performed by Bob. However, this view is incorrect also, and results, as Einstein, Podolsky and Rosen recognized, in deep inconsistencies when combined with relativity theory. It is possible to set up the EPR scenario so that one observer sees Alice measure first, then Bob, while another observer sees Bob measure first, then Alice. According to relativity, physics must equally well explain the observations of the first observer as the second. While our terminology of cause and effect cannot be compatible with both observers, the actual experimental values are invariant under change of observer. The experimental results can be explained equally well by Bob's measuring first and causing a change in the state of Alice's particle, as the other way around. This symmetry shows that Alice and Bob cannot, in fact, use their EPR pair to communicate faster than the speed of light, and thus resolves the apparent paradox. All that can be said is that Alice and Bob will observe the same random behavior.

In the following sections on teleportation and dense coding, we show how EPR pairs can be used to aid communication.

II. QUANTUM TELEPORTATION

Before we look into quantum teleportation, we review the No Cloning Theorem that states that an unknown quantum state can not be perfectly cloned.

A The No Cloning Theorem

The unitary property implies that quantum states cannot be copied or cloned. The no cloning proof is a simple application of the linearity of unitary transformations.

Assume that U is a unitary transformation that clones, in that $U(|a0\rangle) = |aa\rangle$ for all quantum states $|a\rangle$. Let $|a\rangle$ and $|b\rangle$ be two orthogonal quantum states. Say $U(|a0\rangle) = |aa\rangle$ and $U(|b0\rangle) = |bb\rangle$. Consider $|c\rangle = (1/\sqrt{2})(|a\rangle + |b\rangle)$. By linearity,

$$\begin{aligned} U(|c0\rangle) &= \tfrac{1}{\sqrt{2}}(U(|a0\rangle) + U(|b0\rangle)) \\ &= \tfrac{1}{\sqrt{2}}(|aa\rangle + |bb\rangle). \end{aligned}$$

But if U is a cloning transformation then

$$U(|c0\rangle) = |cc\rangle = 1/2(|aa\rangle + |ab\rangle + |ba\rangle + |bb\rangle),$$

which is not equal to $(1/\sqrt{2})(|aa\rangle + |bb\rangle)$. Thus there is no unitary operation that can reliably clone unknown quantum states. It is clear that cloning is not possible by using measurement since measurement is both probabalistic and destructive of states not in the measuring device's associated subspaces.

It is important to understand what sort of cloning is and isn't allowed. It is possible to clone a known quantum state. What the no cloning principle tells us is that it is impossible to reliably clone an unknown quantum state. Also, it is possible to obtain n particles in an entangled state $a|0000\ldots0\rangle + b|1111\ldots1\rangle$ from an unknown state $a|0\rangle + b|1\rangle$. Each of these particles will behave in exactly the same way when measured with respect to the standard basis for quantum computation $\{|00\ldots0\rangle, |00\ldots01\rangle, \ldots, |11\ldots1\rangle\}$, but not when measured with respect to other bases. It is therefore impossible to create the n particle state $(a|0\rangle + b|1\rangle) \otimes \ldots \otimes (a|0\rangle + b|1\rangle)$ from an unknown state $a|0\rangle + b|1\rangle$.

B Telportation

Teleportation is a process of reproducing an unknown quantum state by the use classical communication and EPR pairs. The objective is to transmit the quantum state of a particle using classical bits and reconstruct the exact quantum state at the receiver. Since quantum state cannot be copied, the quantum state of the given particle will necessarily be destroyed. Here is the protocol for teleporting a quantum state.

Alice Alice has a qubit whose state she doesn't know. She wants to send the state of ths qubit

$$\phi = a|0\rangle + b|1\rangle$$

to Bob through classical channels. Alice and Bob each possess one qubit of an entangled pair

$$\psi_0 = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

The starting quantum state is written as

$$\begin{aligned}\phi \otimes \psi_0 &= \frac{1}{\sqrt{2}}(a|0\rangle \otimes (|00\rangle + |11\rangle) + b|1\rangle \otimes (|00\rangle + |11\rangle)) \\ &= \frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle),\end{aligned}$$

of which Alice controls the first two bits and Bob controls the last one. Alice now applies $C_{not} \otimes I$ and $H \otimes I \otimes I$ to this state:

$$\begin{aligned}&(H \otimes I \otimes I)(C_{not} \otimes I)(\phi \otimes \psi_0) \\ &= (H \otimes I \otimes I)(C_{not} \otimes I)\frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle) \\ &= (H \otimes I \otimes I)\frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|110\rangle + b|101\rangle) \\ &= \frac{1}{2}(a(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + b(|010\rangle + |001\rangle - |110\rangle - |101\rangle)) \\ &= \frac{1}{2}(|00\rangle(a|0\rangle + b|1\rangle) + |01\rangle(a|1\rangle + b|0\rangle) + |10\rangle(a|0\rangle - b|1\rangle) + |11\rangle(a|1\rangle - b|0\rangle))\end{aligned}$$

where C_{not} is the Controlled-NOT operator, I the identity operator and H is the Hadamard Tranform. Alice measures the first two qubits to get one of $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|11\rangle$ with equal probability. Depending on the result of the measurement, the quantum state of Bob's qubit is projected to $a|0\rangle + b|1\rangle$, $a|1\rangle + b|0\rangle$, $a|0\rangle - b|1\rangle$, or $a|1\rangle - b|0\rangle$ respectively. Alice sends the result of her measurement as two classical bits to Bob.

Note that when she measured it, Alice irretrievably altered the state of her original qubit ϕ, whose state she is in the process of sending to Bob. This loss of the original state is the reason teleportation does not violate the no cloning principle.

Bob When Bob receives the two classical bits from Alice he knows how the state of his half of the entangled pair compares to the original state of Alice's qubit.

bits received	state	decoding
00	$a\|0\rangle + b\|1\rangle$	I
01	$a\|1\rangle + b\|0\rangle$	X
10	$a\|0\rangle - b\|1\rangle$	Z
11	$a\|1\rangle - b\|0\rangle$	Y

Here X,Y, and Z are the Pauli operators. Bob can reconstruct the original state of Alice's qubit, ϕ, by applying the appropriate decoding transformation to his part of the entangled pair. Note that this is the encoding step of dense coding.

III. SUPERDENSE CODING

Superdense coding uses one quantum bit together with an EPR pair to encode and transmit two classical bits. Since EPR pairs can be distributed ahead of time, only one qubit (for example a photon) needs to be physically transmitted to communicate two bits of information. This result is surprising as only one classical bit's worth of information can be extracted from a qubit. Superdense Coding is the opposite of teleportation, in that it uses two classical bits to transmit a single qubit.

The key to both dense coding and teleportation is the use of entangled particles. The initial set up is the same for both processes. Alice and Bob wish to communicate. Each is sent one of the entangled particles making up an EPR pair,

$$\psi_0 = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Say Alice is sent the first particle, and Bob the second. So until a particle is transmitted, only Alice can perform transformations on her particle, and only Bob can perform transformations on his.

A The Superdense Coding Protocol

Alice Alice receives two classical bits, encoding the numbers 0 through 3. Depending on this number Alice performs one of the transformations $\{I, X, Y, Z\}$ on her qubit of the entangled pair ψ_0. Transforming just one bit of an entangled pair means performing the identity transformation on the other bit. The resulting state is shown in the table.

Value	Transformation	New state
0	$\psi_0 = (I \otimes I)\psi_0$	$\frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$
1	$\psi_1 = (X \otimes I)\psi_0$	$\frac{1}{\sqrt{2}}(\|10\rangle + \|01\rangle)$
2	$\psi_2 = (Y \otimes I)\psi_0$	$\frac{1}{\sqrt{2}}(-\|10\rangle + \|01\rangle)$
3	$\psi_3 = (Z \otimes I)\psi_0$	$\frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$

Alice then sends her qubit to Bob.

Bob Bob applies a controlled-NOT to the two qubits of the entangled pair.

Initial state	Controlled-NOT	First bit	Second bit
$\psi_0 = \frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$	$\frac{1}{\sqrt{2}}(\|00\rangle + \|10\rangle)$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	$\|0\rangle$
$\psi_1 = \frac{1}{\sqrt{2}}(\|10\rangle + \|01\rangle)$	$\frac{1}{\sqrt{2}}(\|11\rangle + \|01\rangle)$	$\frac{1}{\sqrt{2}}(\|1\rangle + \|0\rangle)$	$\|1\rangle$
$\psi_2 = \frac{1}{\sqrt{2}}(-\|10\rangle + \|01\rangle)$	$\frac{1}{\sqrt{2}}(-\|11\rangle + \|01\rangle)$	$\frac{1}{\sqrt{2}}(-\|1\rangle + \|0\rangle)$	$\|1\rangle$
$\psi_3 = \frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$	$\frac{1}{\sqrt{2}}(\|00\rangle - \|10\rangle)$	$\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	$\|0\rangle$

Note that Bob can now measure the second qubit without disturbing the quantum state. If the measurement returns $|0\rangle$ then the encoded value was either 0 or 3, if the measurement returns $|1\rangle$ then the encoded value was either 1 or 2.

Bob now applies H to the first bit:

Initial state	First bit	H(First bit)
ψ_0	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	$\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle) + \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)) = \|0\rangle$
ψ_1	$\frac{1}{\sqrt{2}}(\|1\rangle + \|0\rangle)$	$\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle) + \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)) = \|0\rangle$
ψ_2	$\frac{1}{\sqrt{2}}(-\|1\rangle + \|0\rangle)$	$\frac{1}{\sqrt{2}}(-\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle) + \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)) = \|1\rangle$
ψ_3	$\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	$\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle) - \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)) = \|1\rangle$

Finally, Bob measures the resulting bit which allows him to distinguish between 0 and 3, and 1 and 2.

IV. QUANTUM ERROR CORRECTION

One fundamental problem in building quantum computers is the need to isolate the quantum state. An interaction of particles representing qubits with the external environment disturbs the quantum state, and causes it to decohere, or transform in an unintended and often non-unitary fashion. Adding error correction protocols mitigates the effect of decoherence, quantum error correction is going to be a key component of quantum computation and communications algorithms.

On the surface quantum error correction is similar to classical error correcting codes in that redundant bits are used to detect and correct errors. But the situation for quantum error correction is somewhat more complicated than in the classical case since we are not dealing with binary data but with quantum states.

Quantum error correction must reconstruct the exact encoded quantum state. Given the impossibility of cloning or copying the quantum state, this reconstruction appears harder than in the classical case. However, it turns out that classical techniques can be modified to work for quantum systems.

A Characterization of Errors

In the following it is assumed that all errors are the result of quantum interaction between a set of qubits and the environment. The possible errors for each single qubit considered are linear combinations of no errors (I), bit flip errors (X), phase errors (Z), and bit flip phase errors (Y). A general single bit error is thus a transformation $e_1 I + e_2 X + e_3 Y + e_4 Z$. Interaction with the environment transforms single qubits according to

$$|\psi\rangle \rightarrow (e_1 I + e_2 X + e_3 Y + e_4 Z)|\psi\rangle = \sum_i e_i E_i |\psi\rangle.$$

For the general case of quantum registers, possible errors are expressed as linear combinations of unitary error operators E_i. These could be combinations of single bit errors, like tensor products of the single bit error transformations $\{I, X, Y, Z\}$, or more general multi-bit transformations. In any case, an error can be written as $\sum_i e_i E_i$ for some error operators E_i and coefficients e_i.

B Recovery of Quantum State

An error correcting code for a set of errors E_i consists of a mapping C that embeds n data bits in $n+k$ code bits together with a syndrome extraction operators S_C that maps $n+k$ code bits to the set of indices of correctable errors E_i such that $i = S_C(E_i(C(x)))$. If $y = E_j(C(x))$ for some unknown but correctable error, then error $S_C(y)$ can be used to recover a properly encoded value $C(x)$, i.e. $E^{-1}_{S_C(y)}(y) = C(x)$.

Now consider the case of a quantum register. First, the state of the register can be in a superposition of basis vectors. Furthermore, the error can be a combination of correctable error operators E_i. It turns out that it is still possible to recover the encoded quantum state.

Given an error correcting code C with syndrome extraction operator S_C, an n-bit quantum state $|\psi\rangle$ is encoded in a $n+k$ bit quantum state $|\phi\rangle = C|\psi\rangle$. Assume that decoherence leads to an error state $\sum_i e_i E_i|\phi\rangle$ for some combination of correctable errors E_i. The original encoded state $|\phi\rangle$ can be recovered as follows:

1. Apply the syndrome extraction operator S_C to the quantum state padded with sufficient $|0\rangle$ bits:

$$S_C(\sum_i e_i E_i|\phi\rangle) \otimes |0\rangle = \sum_i e_i (E_i|\phi\rangle \otimes |i\rangle).$$

   Quantum parallelism gives a superposition of different errors each associated with their respective error index i.

2. Measure the $|i\rangle$ component of the result. This yields some (random) value i_0 and projects the state to

$$E_{i_0}|\phi, i_0\rangle$$

3. Apply the inverse error transformation $E^{-1}_{i_0}$ to the first $n+k$ qubits of $E_{i_0}|\phi, i_0\rangle$ to get the corrected state $|\phi\rangle$.

Note that step 2 projects a superposition of multiple error transformations into a single error. Consequently, only one inverse error transformation is required in step 3.

C Error Correction Example

Consider the trivial error correcting code C that maps $|0\rangle \rightarrow |000\rangle$ and $|1\rangle \rightarrow |111\rangle$. C can correct single bit flip errors

$$E = \{I \otimes I \otimes I, X \otimes I \otimes I, I \otimes X \otimes I, I \otimes I \otimes X\}.$$

The syndrome extraction operator is

$$S : |x_0, x_1, x_2, 0, 0, 0\rangle \rightarrow |x_0, x_1, x_2, x_0 \bigoplus x_1, x_0 \bigoplus x_2, x_1 \bigoplus x_2\rangle,$$

with the corresponding error correction operators shown in the table. Note that $E_i = E_i^{-1}$ for this example.

Bit flipped	Syndrome	Error correction
none	$\|000\rangle$	none
0	$\|110\rangle$	$X \otimes I \otimes I$
1	$\|101\rangle$	$I \otimes X \otimes I$
2	$\|011\rangle$	$I \otimes I \otimes X$

Consider the quantum bit $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ that is encoded as

$$C|\psi\rangle = |\phi\rangle = \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)$$

, a three qubit entangled state and the error

$$E = \frac{4}{5}X \otimes I \otimes I + \frac{3}{5}I \otimes X \otimes I.$$

The resulting error state is

$$\begin{aligned}
E|\phi\rangle &= (\frac{4}{5}X \otimes I \otimes I + \frac{3}{5}I \otimes X \otimes I)(\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)) \\
&= \frac{4}{5}X \otimes I \otimes I(\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)) + \frac{3}{5}I \otimes X \otimes I(\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)) \\
&= \frac{4}{5\sqrt{2}}X \otimes I \otimes I(|000\rangle - |111\rangle) + \frac{3}{5\sqrt{2}}I \otimes X \otimes I(|000\rangle - |111\rangle) \\
&= \frac{4}{5\sqrt{2}}(|100\rangle - |011\rangle) + \frac{3}{5\sqrt{2}}(|010\rangle - |101\rangle)
\end{aligned}$$

Next apply the syndrome extraction to $(E|\phi\rangle) \otimes |000\rangle$ as follows:

$$\begin{aligned}
&S_C((E|\phi\rangle) \otimes |000\rangle) \\
&= S_C(\frac{4}{5\sqrt{2}}(|100000\rangle - |011000\rangle) + \frac{3}{5\sqrt{2}}(|010000\rangle - |101000\rangle)) \\
&= \frac{4}{5\sqrt{2}}(|100110\rangle - |011110\rangle) + \frac{3}{5\sqrt{2}}(|010101\rangle - |101101\rangle) \\
&= \frac{4}{5\sqrt{2}}(|100\rangle - |011\rangle) \otimes |110\rangle + \frac{3}{5\sqrt{2}}(|010\rangle - |101\rangle) \otimes |101\rangle
\end{aligned}$$

Measuring the last three bits of this state yields either $|110\rangle$ or $|101\rangle$. Assuming the measurement produces the former, the state becomes

$$\frac{1}{\sqrt{2}}(|100\rangle - |011\rangle) \otimes |110\rangle.$$

The measurement has the almost magical effect of causing all but one summand of the error to disappear. The remaining part of the error can be removed by applying the inverse error operator $X \otimes I \otimes I$, corresponding to the measured value $|110\rangle$, to the first three bits, to produce

$$\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle) = C|\psi\rangle = |\phi\rangle.$$

v. CONCLUSIONS

Quantum computing and communications are novel ways of engineering quantum systems and are proving to dramatically change the way we think about computation, complexity, information, and communication. Quantum entanglement is one of the most important consequences of quantum mechanics and ,as we have seen, introduces a new dimension to computation and communications.

VI. REFERENCES

1. R. P. Feynman. Simulating physics with computers, International Journal of Theoretical Physics, **(21)** 467, 1982.

2. A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete, Physical Review, **(47)**, 777-780, 1935.

3. M. A. Nielsen and I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

4. D. Bouwmeester, J. W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger, Experimental quantum teleportation, Nature, **(390)**, 575-579, 1997.

5. J. Preskill, Quantum Compuation and Information, California Institute of Technology, 1998, http://www.theory.caltech.edu/people/preskill/ph229.

6. N. Gershenfeld and I. L. Chuang, Bulk spin resonance quantum computation, Science, **(275)**, 350, 1999.

7. C. H. Bennett and D. P. DiVincenzo, Quantum information and computation, Nature, **(404)**, 247, 2000.

8. E. Schrodinger, Probability relations between separated systems, Proceedings of Cambridge Philosophical Society, **(32)**, 446, 1936.

Self-assembling Nanotube-based Circuits with Collagen Substrates

Sharlene Coleman
Mills College

David Elihu
Massachusetts Institute of Technology

Sarah Frost
University of Notre Dame

Todd Hoffenberg
Georgia Institute of Technology

Ziyad Jabaji
Stanford University

David Karig
Princeton University

Elena Losseva
University of Western Ontario

Don Riggs
Union College

Abstract

We suggest several assembly strategies for creating a collagen substrate capable of supporting a self-assembling carbon nanotube based circuit. Several strategies are explored and their strengths and weaknesses highlighted. A novel structure consisting of yeast-generated flat collagen strands with 'sticky ends' which tile to form a flat lattice structure is proposed as the support for a carbon nanotube circuit. The nanotubes are affixed to the collagen base by a double-sided molecule consisting of a pyrenoid moiety that binds carbon nanotubes and an antibody-functional tail that binds to collagen.

1 Collagen Introduction

Collagen, the most abundant protein in the body, provides structure and resilience to tissues. Collagen has a right-handed triple helical framework, and its composition is characterized by an abundance of two amino acids, glycine and proline. Glycine is essential for the right-handed triple helix structure, and collagen is typically composed of a three amino acid chain that consists of glycine and two other amino acids. Collagen's rigid yet flexible structure makes it a suitable construction material. The ability to choose from the 20 amino acids in the other two amino acid chains of the structure gives additional flexibility in designing specific collagen strands. Similarity to nanotubes in size is another factor in favor of using collagen as a building material for nanocircuits. A typical collagen molecule is 300 nm long and 1.5 nm in diameter, quite similar to the 1.1 nm diameter of a carbon nanotube. Another major advantage of using collagen in the molecular framework is its strength, which is due to the hydrogen bonds chaining the triple helix together. [2][1] Researchers have recently reported electrospinning of collagen nanofibers and in-vitro development of collagen. In order to produce the carefully engineered collagen necessary for this scheme, the best production choice would be using living cells as factories. Yeast is being explored as a collagen producer, and perhaps simpler organisms can be employed in this manner. [3]

Using collagen as the molecular scaffold may hold some potential problems attributable to its molecular framework. At high temperatures, particularly above 40C, the triple-helix unwinds and the collagen molecule denatures. New studies are currently targeting improvements in the thermal stability of collagen. Potentially problematic changes in collagen's strength and rigid structure over time demand further investigation.

2 Nanotube Introduction

Carbon nanotubes are single cylindrical molecules in the fullerene molecular family that exhibit the same closed structure as buckyballs; nanotubes extend into long strands rather than forming a spherical structure. Nanotubes are extremely resistant to stress. They form as single walled nanotubes (SWNT) or multiwalled nanotubes (MWNT). Both SWNT and MWNT share the exciting electrical properties which make them attractive for use in nanoelectronics. The single walled nanotubes are considered here due to their smaller size and simpler structure.

There are three types of nanotubes that differ in the angle at which they are rolled. Depending on this angle, nanotubes can function either as semiconductors or conductors. The types include armchair, zigzag, and chiral. The 'armchair' type exhibits metallic conductivity and resistance that increases with heat [5][19]. The chiral type produces semiconducting nanotubes. The architecture considered later in this paper requires both metallic and semiconducting nanotubes. There are several known production methods such as chemical vapor deposition[7] and arc discharge[12] among others [10]. However, there is no known method for grow-

ing a particular type of nanotube nor for separating one type from another.

3 Collagen Assembly Strategies

3.1 Overlap Strategies

The construction of a self-assembling carbon nanotube lattice with a collagen substrate naturally begins with substrate assembly. In nature, collagen triple helix strands bond together in parallel rows, head to tail, with a gap between the head of one and the tail of the next and an offset between the parallel rows.

With this structure in mind, several design strategies were considered. The first and most obvious was to construct a grid of single collagen strands (see figure 1). The obvious defects in this model are the limited binding surface available where the two strands meet at a 90 degree angle and the existence of gaps between the head of one strand and the tail of the next. Creating a good bond with this structure is problematic because the diameter of a single collagen strand is 1.1 nm, providing a limited surface area for bonding.

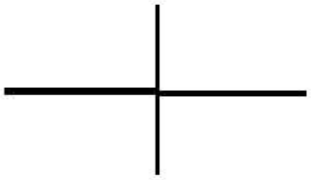


Figure 1. Strategy 1: Simple crossover

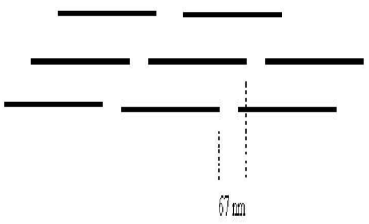


Figure 2. Head to tail collagen assembly illustrating gaps and offsets. Note: Not to scale

A second strategy seeks to take advantage of the offset when parallel collagen strands bond to one another. See figure 2. In this strategy, junctions present at the parallel fiber overlaps would contribute significantly to the strength of the resulting structure by providing multiple points of contact for the perpendicular strands, and the presence of multiple strands eliminates the potential problem of gaps between the head of one fiber and the tail of the next. Two types of collagen strands would be required, one with suitable binding sites for nanotubes and the other without (figure 3).

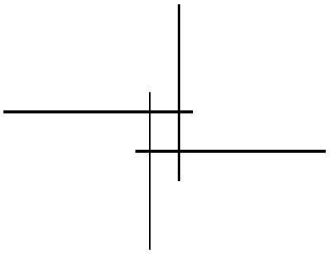


Figure 3. Strengthen structure with multiple overlaps

This strategy can be scaled up to include multiple strands with proportionally greater contact area, perhaps with alternating types of collagen strands (figure 4).

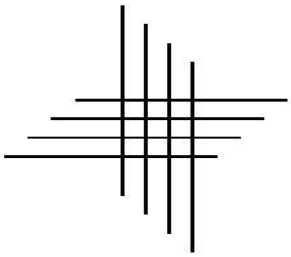


Figure 4. Scale up multiple overlap approach

Collagen strands designed to form a single 90 degree bend might lend themselves to an assembly strategy using tiling (figure 6).

One of the ways to construct regular grids out of collagen is to join collagen triple helices to form the vertices of a grid. Several grid types can be envisioned, depending upon how many helices can be joined in a knot (figure 7). Note that it is not possible to join 3 helices, since the number of individual strands to be joined is odd. It is possible, however, to design a tile containing a node of 4 collagen helices, which, if assembled into a sheet would form a regular grid.

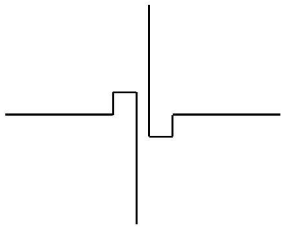


Figure 5. Increase binding surface area

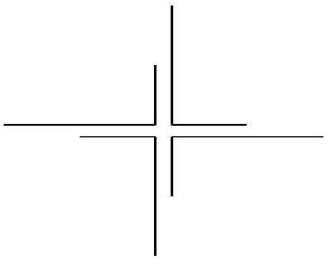


Figure 6. Tiling with single 90 degree bends.

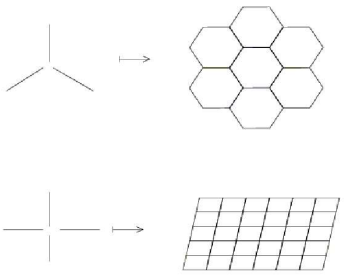


Figure 7. Grid configurations considered.

The design of a tile is governed by two major factors: strand orientation within each helix must coincide and it should be possible to arrange tiles side by side as to cover the plane. The first of the following tiles (figure 8) has the problem that the three sequences in each helix do not have the same orientation. The second tile cannot be used either because the direction of the helices do not allow for sequential tiling. The last of these tiles permits the construction of a grid as shown in figure 3.1. Note that the junction needs to be flat in order to avoid rigidity problems.

As an aside, one might reasonably question whether this structure already exists in nature. If it could be constructed in vitro, a binding antibody could be found and employed to search for the identical structure in vivo.

The first step in the self-assembly of the collagen grid is

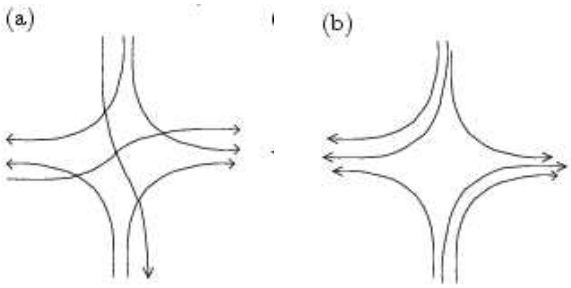


Figure 8. Tiles A and B

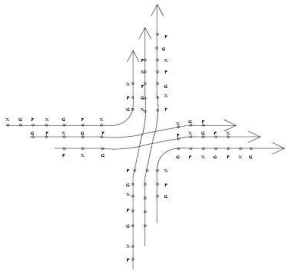


Figure 9. Tile C

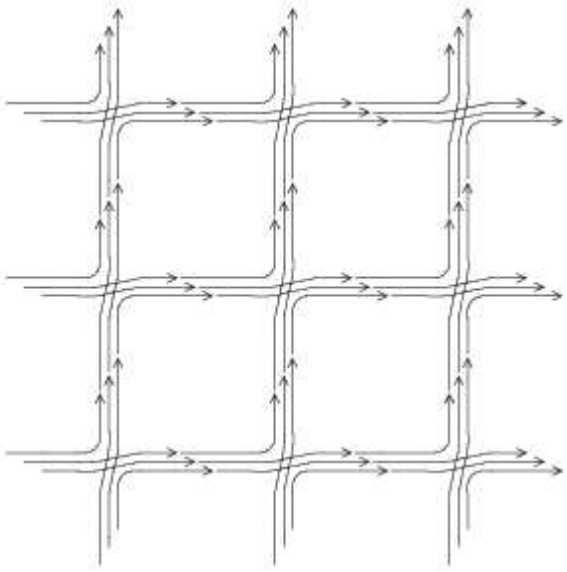


Figure 10. Grid made of type C tiles

to engineer the strands needed to form a tile. Yeast cells have demonstrated the expression of collagen. Using bacterial culture for the manufacturing of collagen would be optimal because of the possible simplicity of the cell which would al-

low better control over the production process. Current DNA production techniques must be extended for specifically engineering collagen. Care must be taken to ensure that there is a unique way for the engineered strands to bind to each other (six are required). This can be achieved by using appropriate amino acids to differentiate all binding sites. In order to construct the collagen lattice, the utilization of a seed tile as well as edge tiles will be required. As in DNA-tiled structures, an assembled sheet may be prone to errors involving tiles twisting and binding incorrectly if tiles are not sufficiently rigid (for discussion of tiling see for instance [17][9]). A reinforcing mechanism may be necessary to provide additional rigidity.

3.2 Molecular Lashing

Molecular lashing, utilizing a molecule such as rotaxane to attach to and reinforce the juncture, is an example of such an approach. Such a scheme involves searching for molecules that could bring about binding equivalent to rope lashing. The molecule pictured in figure 11 is a didactic example that demonstrates the likely nature of the four "arms" desired for reinforcing a four-way collagen junction, in which two such molecules come together and fuse over the collagen intersection.

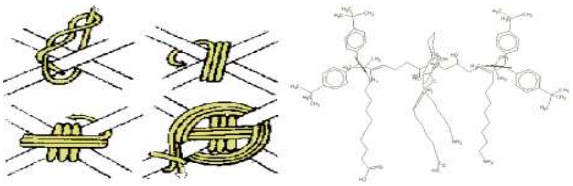


Figure 11. Conventional (diagonal) rope lashing and a molecule that could (theoretically) do the equivalent when two come together and react over the intersection.

4 Glues

4.1 Antibodies

Various possibilities exist for attaching nanotubes to collagen. Antibodies, which are "Y-shaped" immunoglobulins that bond to epitope sites on antigens, are a useful tool. An epitope site is a sequence of typically six or fewer amino acids. Bonding between antibodies and antigens is non-covalent [13]. Figure 12 shows a typical antibody, which consists of two heavy chains and two light chains. The variable region is the portion of the antibody that binds to the antigen. Recall that proteins can be conjugated to antibodies and that secondary antibodies can bind to the constant regions of other antibodies.

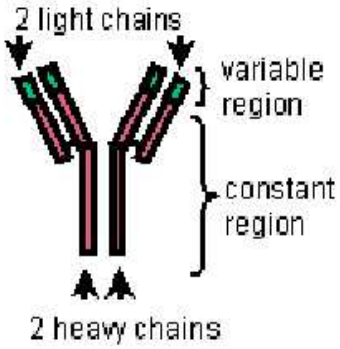


Figure 12. Typical antibody

In order to avoid interfering with the electrical properties of the nanotubes, no covalent bonds may come in contact with them. A team at Stanford presents a non-covalent method for immobilizing biomolecules on the surfaces of carbon nanotubes [4]. This method creates several options for attaching nanotubes to collagen, and since the bonds between antibodies and antigens are non-covalent, antibodies could also be directly attached to nanotubes without affecting their behavior. Two examples have been shown in Figure 13 to illustrate possibilities for attaching nanotubes to collagen. In the first example, the method presented in [4] would be used to immobilize streptavidin on the surface of nanotubes. Streptavidin is a tetramer capable of irreversibly binding biotin molecules. Thus, a biotin conjugated antibody that binds to collagen can be used to link collagen to the streptavidin on the nanotube. In the second example, ferritin is immobilized on the nanotube surface. A protein having both a binding site for ferritin and a binding site for collagen could be engineered, possibly by linking two antibodies, to attach the nanotube to the tile. Other options are available [8][16][20][18].

4.2 Nanotube Functionalization

We must address the question of how to functionalize – and hence bind, manipulate, and assemble – nanotube molecular wires. The problem at hand is not to be considered within the paradigm of traditional chemical functionalization, as we have to be exceedingly careful not to disrupt the delocalized electronic system in each nanotube molecule. One extremely attractive solution consists of employing the molecule pictured in figure 14, with delocalized aromatic groups that can bind non-covalently and irreversibly to nanotubes, providing an ideal ligand that can easily be attached to collagen substrates through traditional chemical synthesis

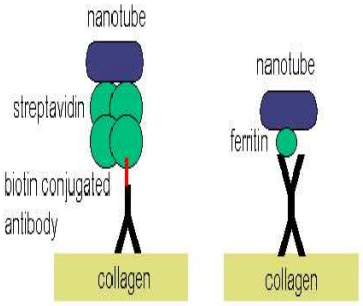


Figure 13. Possibilities for attaching nanotubes to collagen

and reactivity.

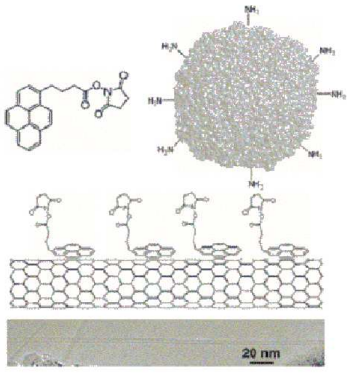


Figure 14. Structure of the ligand (1-Pyrenebutanoic Acid, Succinimidyl Ester) used by the Dai group at Stanford in order to functionalize nanotubes and anchor proteins[4].

5 Architecture

From the standpoint of building useful circuits out of the established molecular toolkit, the circuit architecture becomes very limited. Only single-walled nanotube (SWNT) intersections will be considered. Charles Lieber (et al.) has demonstrated a termed electrostatic switch using nanotubes, which requires nothing more than a SWNT cross-point. If placed with lithography techniques, the nanotubes will separate in the z-dimension to approach a mechanical minimum. The resistance between these nanotubes is finite and very large. However, if the nanotubes are held at opposite electrical polarities, they can be effectively doped in a non-volatile but still reversible manner. This will cause the nanotubes to attract, and the z-dimension distance between them will be held at a van der Walls minimum (ON state). The resistance here is still finite, but it is orders of magnitude less than the mechanical minimum (OFF state). Both switch states are shown in Figure 15.

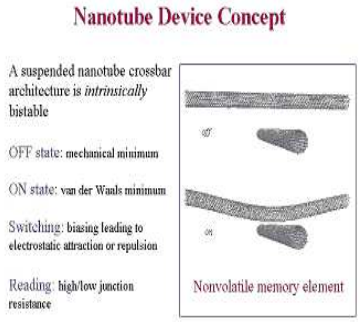


Figure 15. Electrostatic switch

The resistance difference between states spans 5 orders of magnitude. It is important to note that this device can be made with any combination of semiconducting and metallic nanotubes; however, implanting diode characteristic (essential for wired logic) into an intersection requires a lower layer of semiconducting wires (not necessarily nanotubes) and an upper layer of metallic nanotubes [11] [14]. Unfortunately, there is no current way of separating semiconducting from metallic nanotubes, but it must be presumed this will soon be possible. To construct a RAM, which could be done at present, a crossbar grid is most convenient. To be defect tolerant, a good addressing scheme must be considered. On a nano-scale, work done by Phil Kuekes HP Labs and separately by Andre DeHon [6] proposes solutions to addressing large crossbar architectures. More importantly, this applies to logic. If a 5 nm separation between parallel nanotubes is achieved, informational density is comparable to $10^12/cm^2$, and switch speed can approach 200 GHz [11]. For large-scale crossbar logic architecture and a demonstration of fault tolerance, the Teramac can be considered a reference point [15]. Because the scaffolding is not intolerant to heat, power dissipation must be considered. If each device operates as Lieber outlines, a 0.3V supply and 3 microA current per device is reasonable [11]. Applying this to one device at each row continually operating at one time, the overall power consumption is near 3W, which may eventually become significant. Additionally, this architecture is susceptible to transient failure from radiation or, potentially, other external effects. Error correction and detection may be difficult to accomplish because the structure is intentionally left as simple as possi-

ble for tiling reasons. It should also be noted that the stress on a nanotubes in the ON state is within bounds, but additional stress in the tiled product may become a factor.

6 Other

6.1 Final Assembly

The creation of a collagen lattice and a method of attaching carbon nanotubes to the collagen structure has been presented. The final step involves positioning a second layer of nanotubes,perpendicular to the first, to form a crossbar architecture

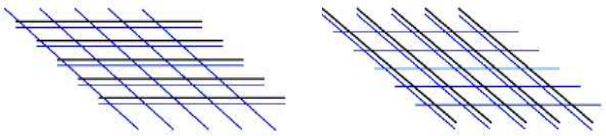


Figure 16. X and Y sandwich components

These layers are combined so the two nanotube layers are both inside the collagen layers which would be secured together with a molecule that bonded to each of the collagen lattices. This molecule would need to be designed carefully to control the distance between the nanotube layers. The two layers need to be near enough to allow the formation of switches at the nodes.

6.2 Future Possibilities

Recall why collagen has been suggested as a material for the nanotube support structure. Precise collagen strands can be manufactured by cells, and the resulting structures will be quite rigid and strong. The collagen used in this lattice will be carefully engineered. Knowledge of the makeup of collagen and the form which it should ultimately take, i.e. a lattice, along with mechanisms available in nature for repairing and replacing collagen can be exploited. Since it is known what the structure should be, a mechanism can be designed to recognize faults in this structure and mark them for replacement or repair. Since the collagen will be on the outside of the nanotube ”sandwich,” perhaps organisms can be created that will actively seek out these collagen lattice faults. While faults and errors are expected in a self-assembled structure, the existence of an active fault detection and correction mechanism may decrease the number of permanent faults and increase the overall lifetime of a system by repairing the lattice when it begins to succumb to natural predators in the environment such as stray particles and unwanted chemical reactions. This self-repairing property will demand a more versatile architecture adapted to these intermittent faults.

7 Acknowledgement

The authors would like to extend a special thanks to Tom Knight for the initial inspiration and generously spending several hours discussing ideas and possibilities.

References

[1] *Molecular Cell Biology*. W.H. Freeman and Co., 1999.

[2] *Lehninger Principles of Biochemistry*. Worth Publishing, 2000.

[3] N.J. Bulleid, D.C.A. John, and K.E. Kadler. Recombinant expression systems for the production of collagen. *Biochemical Society Transactions*, 2000.

[4] R.J. Chen, Y. Zhang, D. Wang, and H. Dai. Noncovalent sidewall functionalization of single-walled carbon nanotubes for protein immobilization. *J. Am. Chem. Soc.*, 2001.

[5] Chin Li Cheung, Andrea Kurtz, Hongkun Park, and Charles M. Lieber. Diameter-controlled synthesis of carbon nanotubes. *J.Phys.Chem B*, 2002.

[6] Andre Dehon. Array-based architecture for molecular electronics. In *First Workshop on Non-Silicon Computation*.

[7] M. Jun, K.Y. Eun, J.K. Lee, Y.J. Baik, K.R. Lee, and J.W. Park. Growth of carbon nanotubes by chemical vapor deposition. *Diamond and Related Materials*, 2001.

[8] A.N. Kirschner, B.F. Erlanger, and S.R. Wilson. A biosensor for fullerenes and carbon nanotubes. In *Eighth Foresight Conference on Molecular Nanotechnology*.

[9] T.H. LaBean, H. Yan, J. Kopatsch adn F.R. Liu, E. Winfree, J.H. Reif, and N.C. Seeman. Construction, analysis, ligation, and self-assembly of dna triple crossover complexes. *Journal of the American Chemical Society*, 2000.

[10] C.N.R. Rao, B.C. Satishkumar, A. Govindaraj, and M. Nath. Nanotubes. *ChemPhysChem*, 2001.

[11] T. Rueckes, K. Kim, E. Joselevich, G.Y. Tseng, C.-L. Cheung, and C.M. Lieber. Carbon nanotube-based nonvolatile random access memory for molecular computing. *Science*, 2000.

[12] Y Saito. Carbon nanotubes produced by arc discharge. *New Diamond and Frontier Carbon Technology*, 1999.

[13] Chemicon http://www.chemicon.com/resource/ANT101/atoc.asp. Introduction to anitbodies. *website*, accessed July 2002.

[14] http://www.cmliris.harvard.edu/html_natalya/research/device/device.html. *website*, accessed July 2002.

[15] http://www.hpl.hp.com/personal/Bruce_Culbertson/TeramacBib.html. *website*, accessed July 2002.

[16] Prozyme http://www.prozyme.com/technical/sa10data.html. Biotin binding proteins. *website*, accessed July 2002.

[17] E. Winfree. Algorithmic self-assembly of dna: Theoretical motivations and 2d assembly experiments. *Journal of Biomolecular Structure and Dynamics*, 2000.

[18] S.S. Wong, E. Joselevich, A.T. Woolley, C.L Cheung, and C.M. Lieber. Covalently functionalized nanotubes as nanometresizedprobes in chemistry and biology. *Nature*, 1998.

[19] Boris I. Yakobson and Richard E. Smalley. Fullerene nanotubes: $C_{1,000,000}$ and beyond. *American Scientist*, 1997.

[20] A. Yazdani and C.M Lieber. Up close and personal to atoms. *Nature*, 1999.

Using Biological Systems as an Analog-to-Digital Converter

David Lee, Robert Lin, Yolanda Zhang

Abstract

Biological systems can create complex structures from very simple systems. To do this, there must be a method to differentiate different regions where identical systems create different structures, such as the abdomen and the head of a fruit fly. Because the differentiation methods in real biological systems are not fully understood, we decided to create a differentiation method based on an analog-to-digital converter using bacterial cells in a concentration gradient. By creating a biological circuit in the bacteria that act as threshold detectors, we can determine the concentration of a chemical at a given distance away from the point source. Then using another biological circuit, we can produce a new concentration gradient that has twice the frequency. If 1 is represented by a concentration of the chemical within the threshold, and a 0 is represented by a concentration outside the threshold, then we can represent any two digit binary number. Thus, we can differentiate separate regions at certain distances away from a point source.

1.0 Introduction

Biological systems have the immense capability to generate complex structures from very simple systems. With simple rules and few inputs, a biological system can grow from a single cell to a multicellular organism in a relatively short amount of time. The sophisticated underlying framework of these systems is extremely powerful, but unfortunately is also beyond anything we can construct with our current technology. However, instead of constructing a complex system, we can tailor the biological system to fit our needs. To show that these systems can be useful, we will build a biological analog to digital converter.

There are many applications for this technology. The scale of technology has been steadily decreasing, and conventional methods of fabrication and production are facing obstacles that are insurmountable. Physical manipulation of substrates will no longer be able to satisfy our need for nanoscale accuracy. Biological systems, however, have been accurately synthesizing nanoscale machines for millions of years. By studying the genetic regulatory pathways and the interactions between proteins, we hope to gain an understanding from an engineering point of view that will allow us to progress towards fully exploiting the potential of biological systems in an engineering process.

2.0 Background

The Drosophila (*Drosophila melanogaster* or the Fruit Fly) is one of the most widely studied systems in terms of its development. The larvae have been studied extensively in an attempt to understand the complex cellular differentiation processes that give rise to the various body parts and organs. Even though we have only very basic understanding of these processes, some of the fundamental ideas have been elucidated from the numerous studies performed.

Segmentation, the development of segments in the body of the larvae, is perhaps the most significant process because the underlying structure of the fly is defined at this stage. However, the input for this complicated process is not complicated at all. The initial input consists of only two inputs: two chemical gradients that have sources located at opposite ends of the larvae [1]. That is, one gradient comes from the anterior end, and the other comes from the posterior end. Transcription factors that are already present in the larvae respond to the varying concentrations of these two signaling chemicals and begin the differentiation process by initiating the transcription of the first level control

genes. These gap genes then in turn activate or repress the expression of what is known as the pair-rule genes.

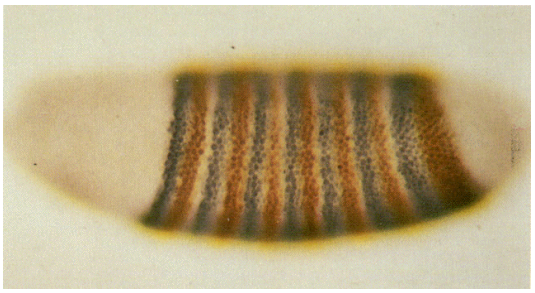


Figure 1: A photograph of the staining of two gene products in a Drosophila larva. The distinct pattern of the two different proteins can be easily seen. (Source: P. A. Lawrence. *The Making of a Fly: The Genetics of Animal Design.* Blackwell Science Inc, Boston, MA: 1992)

The pair-rule genes have a very interesting expression pattern. These pair-rule genes are not expressed in a conventional gradient pattern, but rather in an alternating pattern with distinct borders (see Fig. 1). We decided to study this pattern of expression as the basis for our project in designing a biological analog-to-digital converter.

However, when we investigated this system further we realized that this is not an ideal system for us to manipulate. For each of the stripes present in the larvae, there is a unique locus that controls the expression at that particular stripe. That is, to generate the expression pattern in figure 1, there are fourteen regions in total that are involved in the control of this particular pattern. This is simply too complicated and impractical for us to attempt to manipulate. The problem with scaling is one of the many challenges that this system cannot overcome. With each location requiring a singular control region, as the desired number of stripes increase, the number of controlling regions grows linearly. We decided to engineer a simpler solution for our construction of the biological analog-to-digital converter.

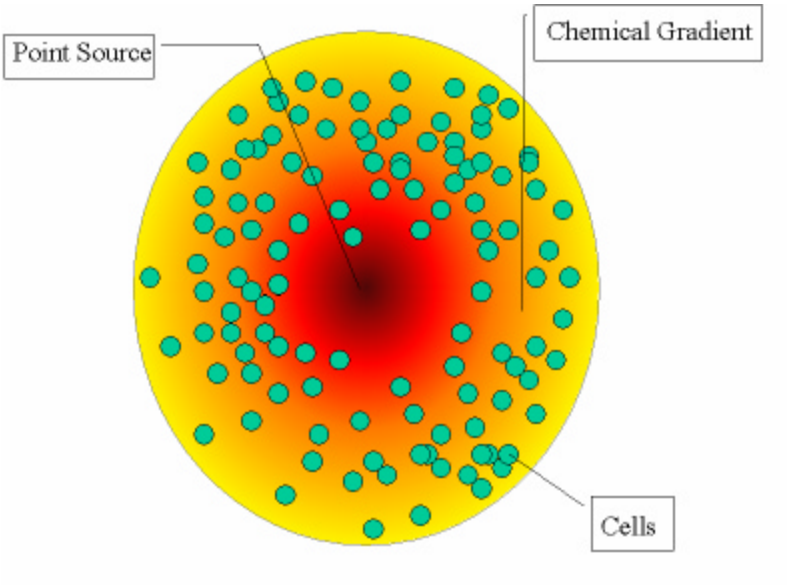


Figure 2: E. coli Cells in a point source chemical gradient

3.0 Bacterial Cells in a Concentration Gradient

3.1 The Setup

For this biological system, *Escherichia coli* bacteria cells on a set plane are introduced to a point-source chemical gradient (see Fig. 2). The chemical causes the cells to produce a protein when the concentration is above a certain threshold. The goal is to produce a sigmoidal curve such that two states, specified 0 and 1, are distinguishable [2].

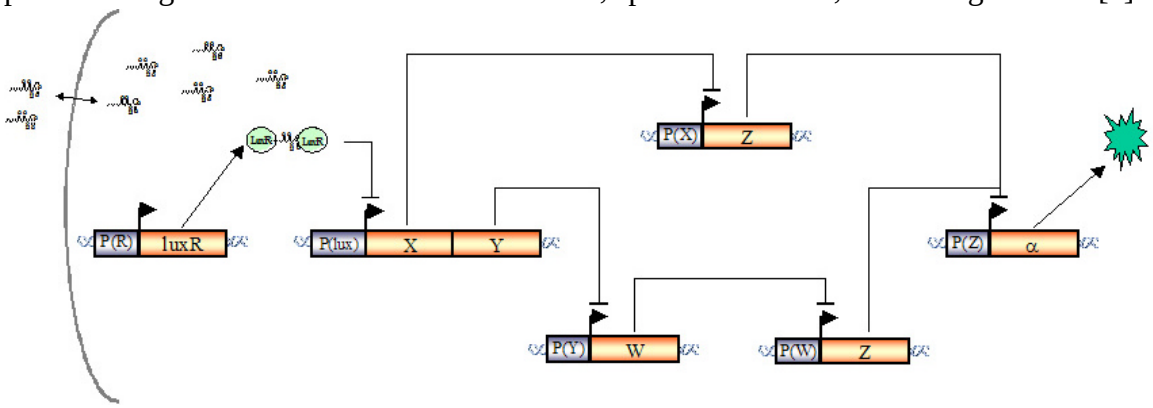


Figure 3: Genetic Circuit for Chemical Concentration Band Detector
(Source: R. Weiss. Cell-Cell Communication. In *Computing Beyond Silicon Conference*, Pasadena, CA, July 2002)

Each E. coli cell is engineered to be a chemical band detector based on Ron Weiss's work. In general, the genetic circuit detects the concentration of a chemical within a specified range and produces a new protein. The mechanism for the circuit is as follows: Chemical 1 diffuses from the point source into the cells and binds with a cytoplasmic protein, allowing the protein to dimerize (see Fig. 3) [3]. The dimer binds to the DNA and activates the P(R) promoter, which produces protein X and protein Y. Protein X will then bind to P(X) and repress the production of protein Z_1. Protein Y will bind to P(Y) and repress the production of protein W, a protein which represses the production of Z_2 at the P(W) site. The production of Z_1 sets the lower threshold and Z_2 sets the higher threshold. When the concentration of chemical 1 is within the threshold region, protein α is produced. The concentration of protein α will produce a sigmoidal curve.

From the concentration of protein α and chemical 1, another genetic circuit (gradient-creation circuit) in the E. coli cell creates a new concentration gradient of chemical 2 with twice the frequency of the sigmoidal curve. Ideally, the concentration of chemical 2 equals the concentration of Chemical 1 subtracted by the concentration of protein α ([C2] = [C1] – [Pα]). To achieve this, protein α must inactivate chemical 1 at a linear rate. Chemical 1 then dimerizes and binds to P(S), which produces chemical 2.

From chemical 2, a similar genetic circuit based on the chemical band detector will produce protein β. Similar to the interaction between protein α and chemical 1, Protein β inactivates chemical 2 to create the next concentration gradient with double the frequency of the expression of β. As long these main circuit elements are created for each sigmoidal curve, the bacterial cells in a concentration gradient can produce more sigmoidal curves with double the frequency of the previous level.

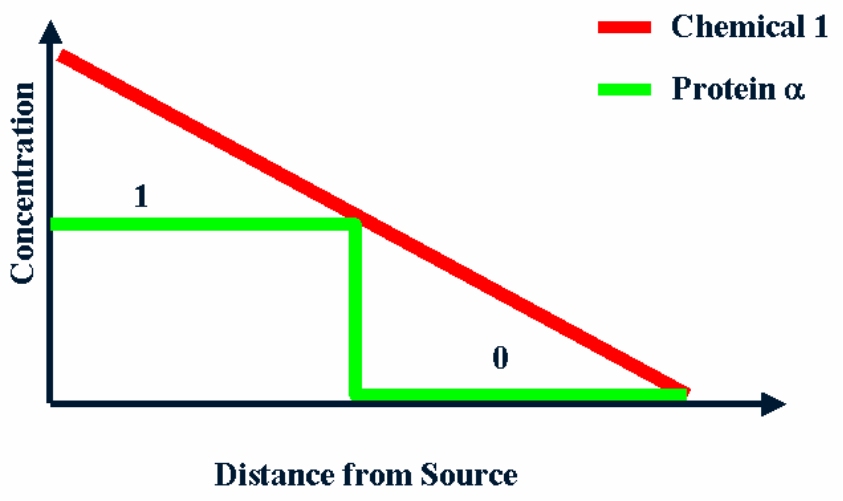


Figure 4: Approximation of a sigmoidal curve (green curve) from an approximated chemical gradient (red curve)

3.2 Creating the Digital Abstraction from Analog Signals

From the basic genetic circuits described above, a bacterial cell can be programmed to produce the sigmoidal curves necessary in our A-D converter based on a simple threshold detector. The importance of the sigmoidal curve is that it provides a sharp transition between two states (see Fig. 4). Because of the sharp transition, the two states, 0 and 1, can be easily distinguished (see Fig. 4). The presence of two distinct states allows for a straightforward digital abstraction from the sigmoidal curves.

Starting with an initial concentration gradient of chemical 1 from the point source, the bacterical cells create a sigmoidal concentration curve of protein a using the band detection circuit (see Fig. 4) (refer section 3.1). A gradient-creation circuit then utilizes both chemical 1 and protein a to produce a new concentration gradient of chemical 2 (see Fig. 5). Using a second band detection circuit that will detect the presence of chemical 2, the bacterial cells create a sigmoidal concentration curve of protein ß which has double the frequency of protein a's curve (see Fig. 6).

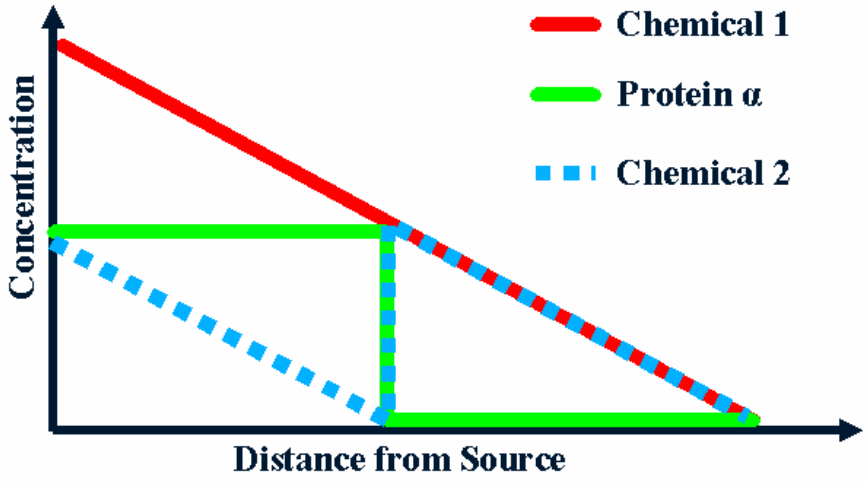


Figure 5: Creating a concentration gradient of chemical 2 from chemical 1 and protein a. Chemical 2 = Chemical 1 - Protein a

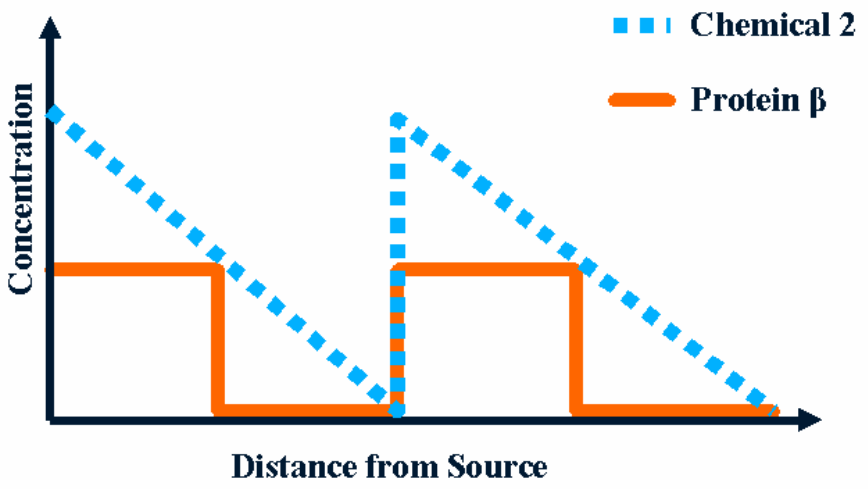


Figure 6: Creating a sigmoidal curve of protein ß from chemical 2's concentration gradient

Each sigmoidal curve represents a specific digit in a binary number. Curves with a lower frequency represent more significant digits, and curves with a higher frequency represent less significant digits (see Fig. 7). Thus three sigmoidal curves will represent all the 3-digit binary numbers.

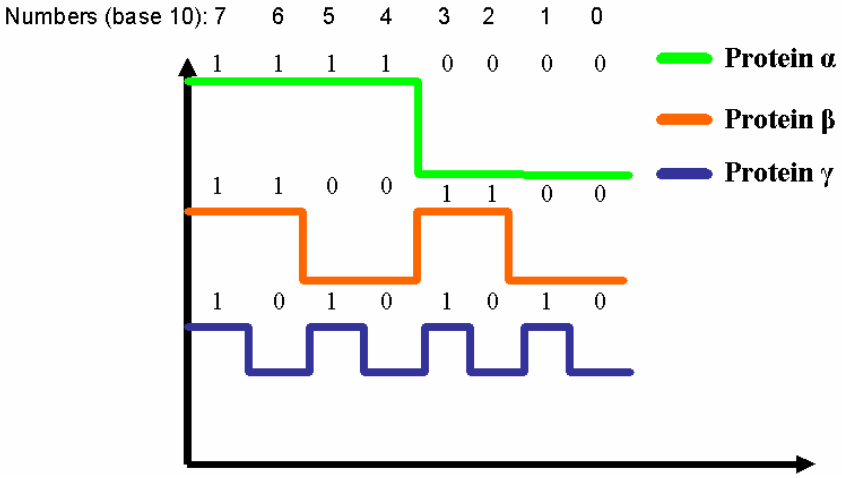


Figure 7: Representation of all three digit binary numbers using the sigmoidal curves

4.0 Work In Progress

With the basic layout for the analog-to-digital converter explained above, it should be possible to build an arbitrarily large converter that can represent any binary number with the necessary number of digits. To do this, there must be a list of all the chemicals and proteins that can be detected by the band detection and the gradient-creation circuits. The chemicals cannot be overly reactive with each other, or else there will be lots of noise in the system, compromising the clarity of the binary representation. Similarly, the proteins cannot counteract each other. The size of the biological A-D converter will thus be limited by this restriction.

The thresholds for the chemical band detection must be determined. If the thresholds are set correctly, we can then have sigmoidal curves that double in frequency between each level. The doubling of frequency is crucial to representing all the possible

binary numbers with a given number of digits in this system. Otherwise, if the thresholds are set incorrectly, the sigmoidal curves will oscillate at unexpected frequencies, causing the analog-to-digital converter to fail to be useful because the resulting curves are undeterministic.

5.0 Summary

We have described a basic framework for building an analog-to-digital converter using a point-source chemical gradient and E. coli bacteria. By designing the bacterial cell to act as a chemical band detector and a chemical gradient creator, we can in theory create an arbitrary large digital system from the analog chemical gradients in the system. We will in reality be limited by the amount of interference between chemicals and proteins, but hopefully with further refinement to the circuits, the amount of interference will be reduced.

References

1. P. A. Lawrence. *The Making of a Fly: The Genetics of Animal Design*. Blackwell Science Inc, Boston, MA: 1992.
2. E. Winfree. Assistant Professor of Computer Science and Computation and Neural Systems. California Institute of Technology. Personal interview. Pasadena, CA. July 5, 2002.
3. S. Basu, D. Karig, and R. Weiss. "Engineering Signal Processing in Cells: Towards Molecular Concentration Band Detection". *Eighth International Meeting on DNA Based Computers*, June 2002.
4. R. Weiss, S. Basu. "The Device Physics of Cellular Logic Gates". *First Workshop on Non-Silicon Computing*, Feb 2002.
5. M. Ptashne. *Genetic Switch: Phage Lambda and Higher Organisms.* Blackwell Science Inc, Boston, MA: 1992.
6. T. Knight. Senior Research Scientist. Massachusetts Institute of Technology. Personal interview. Pasadena, CA. June 28, 2002.

Using Gap Junctions as Basic Building Blocks

Viral Shah* viral@cs.ucsb.edu

Swapnika Ramu‡ sramu@buffalo.edu

August 18, 2002

Abstract

Intercellular communication occurs through exchange of small molecules such as ions, second messengers and small metabolites. This exchange is carried out by clusters of intercellular membrane channels called gap junctions. Structurally, these channels are composed of two subunits, called connexons, which span the plasma membranes of two adjacent cells and interact to form a complete gap junctional channel, thus linking the cytoplasm of one cell to another. We discuss some of the relevant properties of connexins and suggest the use of gap junction channels in nanotechnology and nanoscale circuits.

Contents

*Department of Computer Science, University of California, Santa Barbara

†Department of Biological Sciences, State University of New York at Buffalo

‡Both authors contributed equally to this report.

1 Introduction

Cells communicate with each other in a variety of ways. One such method of communication involves the inter–cellular transmission of molecules through channels in a specialized structure in the cell membrane, the gap junction. Gap junctions provide a regulated pathway linking the cytoplasms of attached cells, and contribute towards ensuring the integration of metabolic activities by setting up networks of directly communicating cell assemblies.

Gap junctions exist between two cells and therefore they differ from other channels in being relatively non-specific and allowing movement of molecules through the channel by passive diffusion. They allow the movement of small molecules such as Ca^{2+} , cyclic AMP and inositol triphosphate. The selectivity appears to be based on molecular size and allows the movement of molecules smaller than 1000 Da but preventing the movement of larger molecules such as proteins and nucleic acids [10].

Co–ordination of cellular activity by gap junctions occurs through a variety of mechanisms which can be classified as follows: speed, synchrony, switching, symbiosis and stimulus/suppression. In the heart, for example, gap junctions permit the rapid cell–to–transfer of action potentials, ensuring coordinated contraction of cardiac muscle cells [1]. In the nervous system, gap junctions allow changes in electric potential to be transmitted from one cell to its neighbors [13]. This is significant because electrical synapses function in neuronal pathways requiring maximum speed, synchronous firing of neurons and rapid switching between neuronal pathways. In non–excitable cells, such as those of the eye, gap junctions allow symbiotic exchanges between highly differentiated, functionally compromised cells and more active, renewable cells. Gap junctions can also act to suppress mutations in key metabolic and signaling enzymes by delivering junction–permeable intermediaries which will allow for the survival of mutant cells with blocked enzymes. This may permit gap junctions to act as tumor suppressors.

1.1 Nomenclature of gap junctions

Connexins are a multi-gene family of proteins, currently believed to have approximately 20 isoforms. The isoforms have approximately 40% overall sequence identity, with much greater identity in the transmembrane and extracellular domains. Some isoforms are restricted to specific species and others exhibit tissue-specificity. Most cells express more than one isoform. These different isoforms

Greek N.	Molecular N.	Mass (kDa)	Organs with Expression
$\alpha 1$	Cx43	43.0	Heart
$\alpha 2$	Cx38	37.8	Embryo
$\alpha 3$	Cx46	46.0	Lens
$\alpha 4$	Cx37	37.6	Lung
$\alpha 5$	Cx40	40.4	Lung
$\alpha 6$	Cx45	45.7	Heart
$\alpha 7$	Cx33	32.9	Testis
$\alpha 8$	Cx50	49.6	Lens
$\beta 1$	Cx32	32.0	Liver
$\beta 2$	Cx26	26.5	Liver
$\beta 3$	Cx31	31.0	Skin
$\beta 4$	Cx31.1	31.1	Skin
$\beta 5$	Cx30.3	30.3	Skin

Table 1: Connexin Multigene Family [10]

compose channels with widely varying unitary conductances, permeabilities, voltage sensitivities and sensitivity to modulatory agents [7].

In invertebrates, gap junctions are formed by a family of proteins known as innexins [11]. Although their gene structure is very different from that of connexins, they remarkably exhibit similar channel properties in terms of regulation by pH, voltage and lipophiles and in terms of permeability.

Two major alternative nomenclatures have been proposed for connexin proteins. One is based on the molecular mass of the polypeptide while the other is based on evolutionary considerations.

1.2 Gap junction structure

Gap junction channels are composed of membrane proteins called connexins. These proteins are organized into hexameric structures known as connexons. An individual connexon on one cell docks with a corresponding connexon on a neighboring cell to form a complete gap junctional channel. Multiple channels then aggregate to form gap junctional plaques. Refer fig 1.

Each connexin has four predominantly hydrophobic, membrane–spanning regions (M1–M4) as shown in fig 2. For Cx43 these domains have been shown to be α–helical. By analogy, the corresponding domains in the other connexins are presumed to be α–helical as well. The carboxy– and amino–domains (CT and NT, respectively) are accessible from the cytoplasm, as is the hydrophilic cytoplasmic loop (CL) domain between M2 and M3. The hydrophilic domains between M1

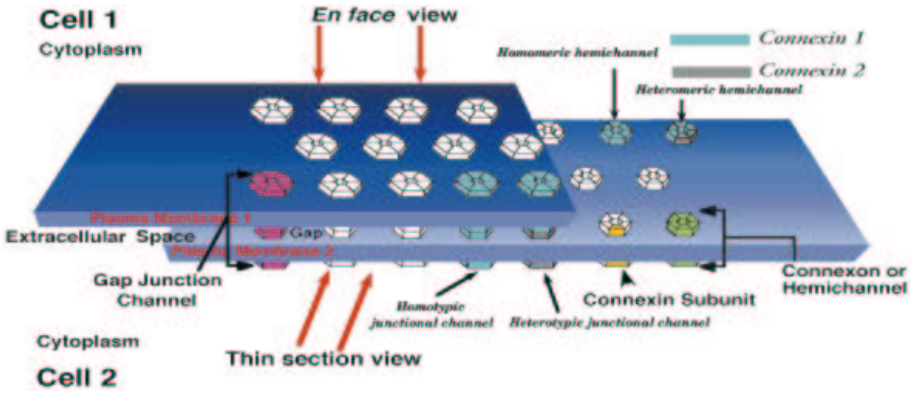


Figure 1: Section view of gap junction channels [7]

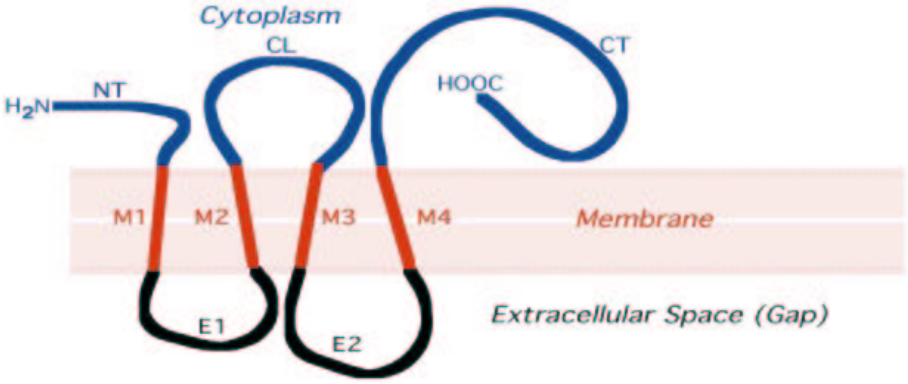


Figure 2: Membrane topology of a connexin monomer [7]

and M2 and between M3 and M4 are accessible from extracellular space and from extracellular loops E1 and E2. The length of the CT varies among connexins and is responsible for most of the differences in molecular mass [7].

The three–dimensional structure of a recombinant gap junction has been determined at a resolution of 7.5 A° in the membrane plane and 21 A° in the perpendicular direction as in Fig 3. The outer diameter of the connexon is $\sim$70 A° and narrows at the extracellular domain to about 50 A° . The channel pore narrows from an apparent $\sim$40 A° diameter at the cytoplasmic side to $\sim$15 A° at the extracellular side of the bilayer and then widens to $\sim$25 A° in the extracellular region. Since amino acid side chains were not visualized at a resolution of 7.5 A° , the functional diameter of the pore is expected to be only 5 A° .

This study also confirms the α–helical nature of the four transmembrane domains of each connexin subunit.In the connexon structure, the α–helices pack in a left–handed bundle, but a single right–handed packing interaction is observed for one of the two possible helix pairs that line the pore. Two of the transmem-

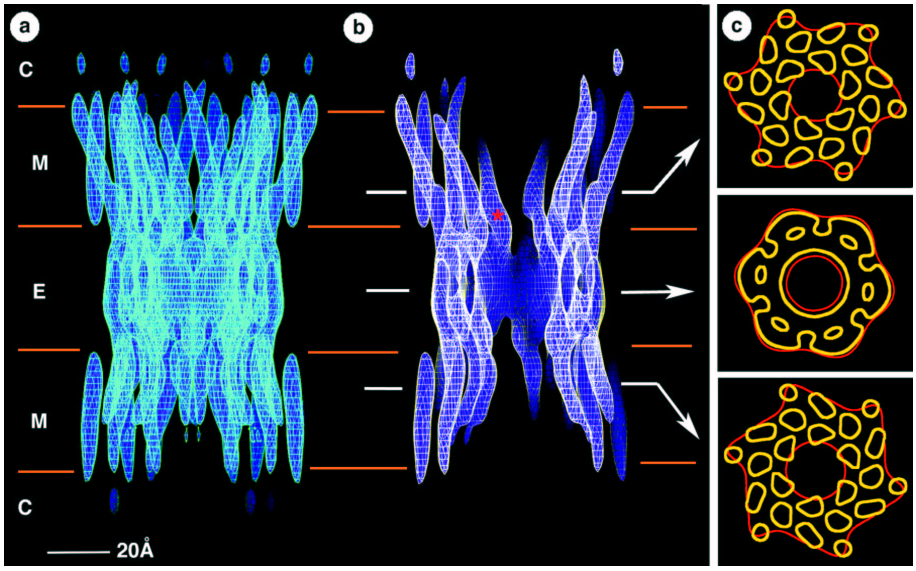


Figure 3: 3 dimensional structure of gap junctions [18]

brane α–helices are strongly tilted and one of these is positioned directly next to the pore. This tilting causes one of the other three helices to be exposed to the channel, creating a criss–cross arrangement of two of the four helices. In addition, computer modeling studies have shown that the formation of an intercellular channel requires a 30° rotation between hemichannels for proper docking.(See [18] and [4]).

1.3 Types of Gap Junction Channels

The different connexin isoforms may associate with each other in many different combinations. The simplest type of subunit is *homotypic*, where the entire connexon is composed of a single connexin. Two such connexons will form a *homomeric* channel. A *heteromeric* connexon is composed of more than one connexin isoform, and two such connexons will dock to form a *heterotypic* channel.

2 Properties

Gap junctions exhibit a variety of properties. We will discuss those which might allow the use of gap junctions as basic building blocks for communications and circuits.

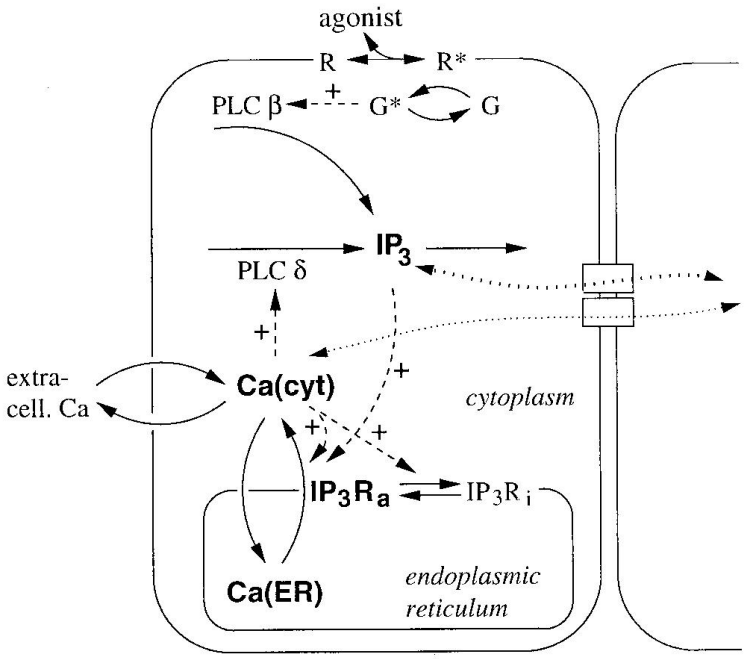


Figure 4: Scheme of the Ca^{2+} and IP_3 dynamics [16]

2.1 Calcium Wave Propagation and Intercellular Communication

Calcium signals provide a mechanism for the integration and transmission of of information. For example, the astrocytes, neurons and glia of the central nervous system exhibit Ca^{2+} signals in response to neurotransmitters such as acetylcholine, noradrenaline, glutamate and adenosine nucleotides. These can travel as intercellular Ca^{2+} waves over several hundred micrometers and may thus provide a basis for a long–range signaling pathway [16]. These waves can propagate through gap–junctional channels.

Gap junction channels allow the diffusion of signaling molecules from cell to cell, thus sustaining the propagating event. These molecules are now generally believed to be inositol triphosphate (IP_3) and Ca^{2+} , although cyclic ADP ribose and other intercellular molecules could fulfill the size requirements and might also trigger regenerative Ca^{2+} mobilization [2]. The neurotransmitters (or agonists) would bind to receptors on the cell membrane, causing activation of the enzyme phospholipase C β. The subsequent rise in IP_3 levels within the cell causes release of Ca^{2+} from intracellular stores (Refer Fig 4). IP_3 can also be generated in downstream cells by PLC activation, thus forming a regenerative step in the propagation mechanism.

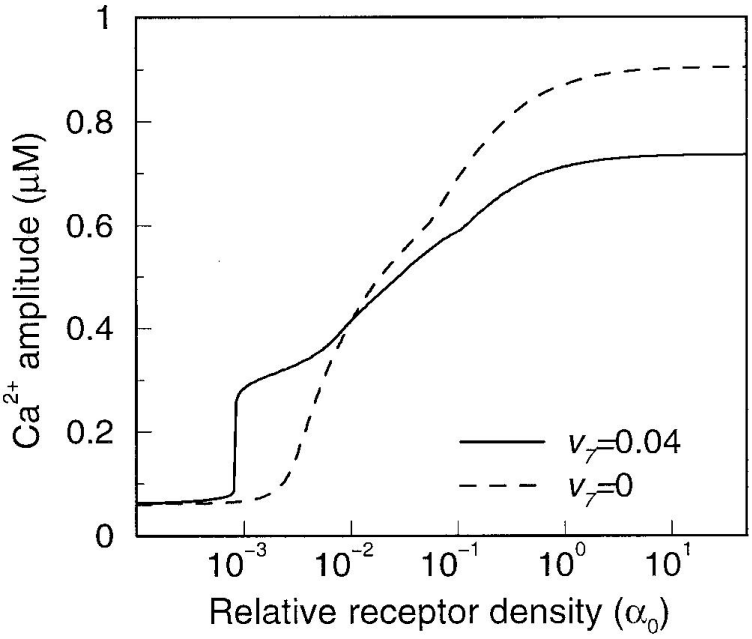


Figure 5: Agonist response in the model of a single cell [16]

Thus intercellular communication can be achieved by controlling the levels of the agonist (activator) which ultimately influences the concentration of Ca^{2+} in the neighboring cells. This behavior is also dependent upon the relative receptor density. It is seen that saturating doses of varying agonists evoke responses of different magnitudes, which may be attributed to differences in overall density and distribution of the receptors for these agonists. In the single–cell model of Hofer et al [16], the Ca^{2+} amplitudes computed for different relative receptor densities reproduce this behavior, as in Fig 5. Similar ideas are also discussed in [12].

Fig 6 depicts the intercellular Ca^{2+} waves. These Ca^{2+} waves could be used to form patterns as in [8] and in Amorphous Computing [5], which is about engineering prespecified, coherent behavior from the co–operation of vast numbers of unreliable parts that are interconnected in unknown, irregular, and time–varying ways.

In [14], intra–cellular genetic circuits are used for cellular communications. Genetic circuits were engineered with the ability to send a controlled signal from one cell, diffuse that signal through the intercellular medium, receive that signal within a second cell, and activate a remote transcriptional response. In contrast to this work, Ca^{2+} wave propagation through gap junctions can be used, which uses non–genetic cellular processes as those described earlier to achieve similar communication. A possible benefit of this method could be that Ca^{2+} wave prop-

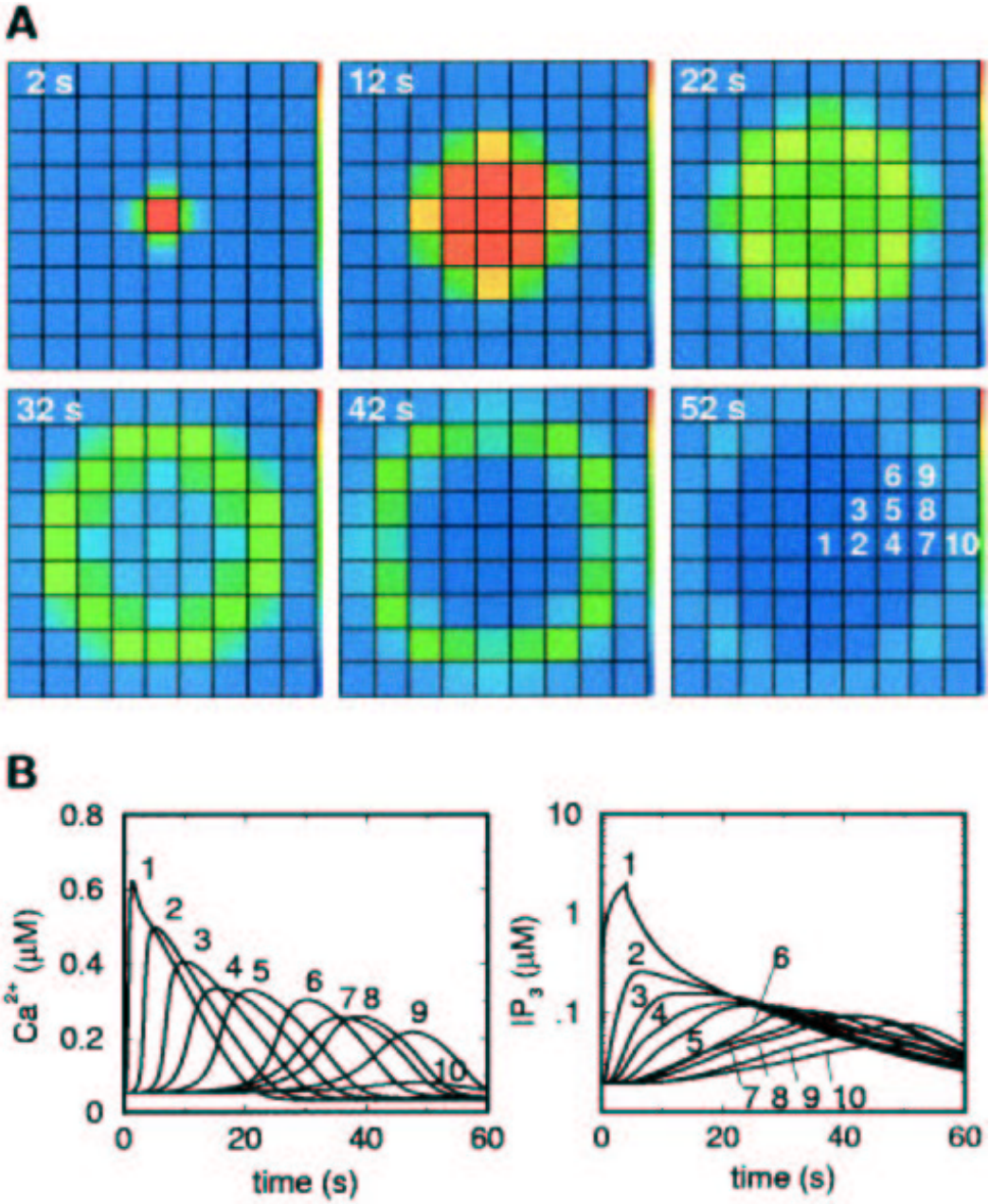


Figure 6: A: Model for agonist evoked intercellular Ca^{2+} waves. The average wave speed in the model for signal propagation through the first three rows of cells is in the same range as the speed computed from experimental data. B: Time courses of [Ca^{2+}] and IP_3 in the cells as labelled in the last frame [16]

agation happens within an order of seconds, whereas genetic circuits might take an order of minutes to react.

2.2 Gating Mechanisms

Gating of a gap junction channel refers to the phenomenon of the opening or closing of the channel due to some stimulus. There are at least three different gating mechanisms associated with gap junction channels—a docking gate, a chemical gate, and a V_j–sensitive gate. [6]

Although the characteristics of voltage–dependent closure of gap junctions have been well studied, the molecular mechanisms involved and even the precise physiological role remain unclear. Channel closure in response to V_j may uncouple a dying cell from its neighbors. Furthermore, mutations that affect voltage gating of Cx32 are linked to a neurodegenerative disease. Knowledge of gating mechanisms is important for identification and classification of connexin proteins as well as in the study of connexin structure and interactions [6].

Connexin channels have several voltage–sensitive processes. The most prominent and well–characterized is the sensitivity of macroscopic junctional conductance to the magnitude of transjunctional voltage (V_j) as shown in fig 7. The conductance is typically maximal at $V_j = 0$ and relaxes symmetrically and slowly (over hundreds of milliseconds to seconds) for hyperpolarizations and depolarizations of either cell [7].

V_j–gating (fig 8 (A,B,C,D))involves rapid voltage–sensitive transitions between a primary open state and one or more subconductance states, but not to a fully closed state. In addition to V_j–gating, a second gating process dependent on junctional voltage is seen in single–channel currents. Unlike V_j–gating, it involves slow transitions into and out of a fully closed state, and is termed *loop gating* [7].

Several connexin mutants with altered voltage sensitivity show a "reversed" response to V_j in that junctional currents activate rather than inactivate when V_j is applied. This "reverse gating" phenomenon was first reported for Cx26 mutants lacking a conserved proline in M2 [6]. The proline is thought to play an essential role in the transduction of a conformational change essential for voltage gating. Amino acid substitutions at a number of sites in M1 of Cx32 influence gating in a similar way.

In [3], it was found that the Cx43 cell–cell channel is not permeable to monovalent positively and negatively charged dyes, which are readily permeable through the fully open channel. These data indicate that a narrowing of the channel pore accompanies gating to the residual state. Consequently, the fast V_j–sensitive gating mechanism can serve as a selectivity filter, which allows electrical coupling but limits metabolic communication.

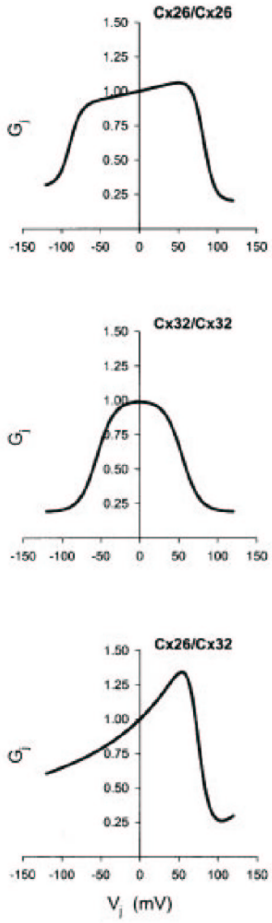


Figure 7: Macroscopic steady state G–V relations for homomeric Cx26, homomeric Cx32 and heterotypic Cx26/Cx32 junctional channels expressed in oocytes [7]

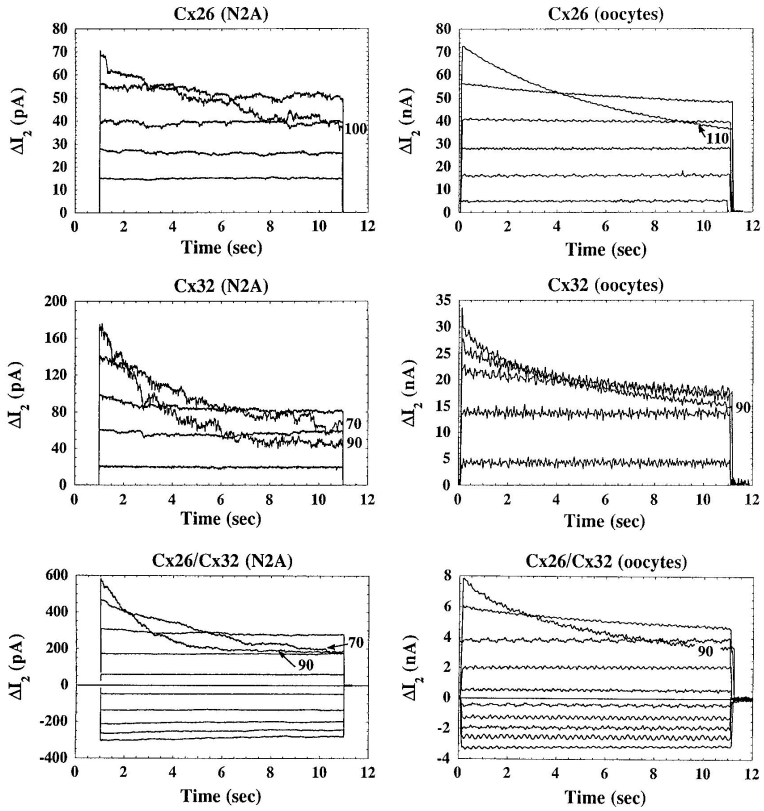


Figure 8: A, B, C and D show voltage gating in homotypic Cx26 and Cx32 in paired N2A cells and Xenopus oocytes. E and F show show voltage gating for the heterotypic channel, and show time– and V_j–dependent decay only for positive V_j steps. [7]

2.3 Rectification

The functional diversity of gap junction intercellular channels arising from the large number of connexin isoforms is significantly increased by heterotypic interactions between members of this family. In [17], a model of electrodiffusion of ions is developed, which predicts that asymmetries in the conductance/permeability properties of hemichannels (refer to fig 7) will produce either an accumulation or a depletion of ions within a channel, depending on voltage polarity, that will result in rectification. Such a property could be exploited while constructing nanoscale circuits.

Table 2: Heterotypic functional compatibilities in mammalian connexins [7]

26	30	30.3	31	31.1	32	33	36	37	40	43	45	46	50	57	
+	+	-	-	-	+		-	-	-	-	-	+	+	-	26
	+	+			+		-	+	+	+	+			-	30
		+	-	-	-		-	+	+	+	-			+	30.3
			+	-	-			-	-	-	-			-	31
				-	-			-	-	-	-			-	31.1
					+	-	-	-	-	-	-	+	+	-	32
						-		-		-					33
							+	-	-	-	-		-		36
								+	+	+	+			+	37
									+	+	+	-	-	-	40
										+	+	+	-	+	43
											+			-	45
												+	+	-	46
													+	-	50
														+	57

Legend: +: forms heterotypic channels, -: does not form heterotypic channels, and blank: not reported.

Group A	Group B
Cx30.3	Cx26
Cx37	Cx32
Cx40	Cx46
Cx43	Cx50
Cx45	
Cx57	

Table 3: Heterotypic functional compatibility groups [7]

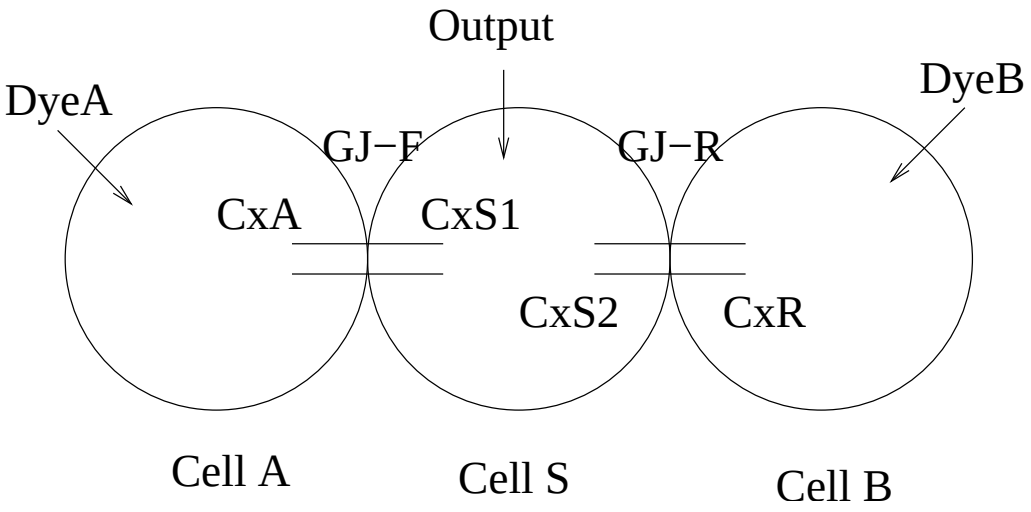


Figure 9: Experimental setup of the 1–bit multiplexer

2.4 Cell–Free Synthesis and Assembly of Gap Junctions

One of the reasons why gap junctions could be attractive in nanotechnology is that it is possible to synthesize and assemble them *in vitro*. In [9], synthesis and assembly of a gap junction connexon was done in microsomal membranes. These were reconstituted into synthetic lipid bilayers. Single channel conductances recorded from these lipid bilayers were similar to those measured for these connexons produced *in vivo*. Further results on synthesis and assembly of connexins *in vitro* into homomeric and heteromeric functional gap junction hemichannels can be found in [15]. However, yields of connexin proteins from *in vitro* assembly are currently low, being about 15 %.

3 Constructing a 1–bit cellular multiplexer

Different connexins can be made to interact in different ways within the cellular environment to produce a combinatorial connexin library. Table 2.4 gives a connexin interaction matrix.

We will now propose a thought experiment to construct a 1–bit multiplexer using gap junction channels, as shown in fig 9. Our setup consists of 3 cells C_A, C_S and C_B. We will call cell C_S the select cell and the cells C_A and C_B the input cells. The output of the multiplexer will also be in C_S. C_S expresses 2 connexins, one showing normal gating behavior and one showing reverse gating behavior. We will label the connexin with normal gating behaviors as CxF, CxS1, CxS2 and the one with the reverse gating behavior as CxR. C_A expresses a connexin CxF

such that it docks only with CxS1 and not with CxS2. C_B expresses a connexin CxR such that it docks only with CxS2 and not with CxS1. A desirable property for CxS1 and CxS2 would be that they form distinct plaques on the cell surface. The gap junction between C_A and C_S is GJ_F and the gap junction between C_S and C_B is GJ_R. The input at C_A is specified by a certain dye dye$_A$ and that at C_B is specified by a dye dye$_B$.

In the resting state, GJ_R is open, and there is transfer of dye dye$_B$ from C_B to C_S. When C_S is raised to a higher potential as compared to C_A and C_B, GJ_R begins to close. On the other hand GJ_F begins to open and transfer of dye dye$_A$ from C_A to C_S should be observed.

Hence, by setting the state of C_S, we have been able to successively select dye transfer from C_A or C_B, and thereby demonstrated the working of a 1–bit cellular multiplexer.

Selective formation of heterotypic channels in mammalian connexins is documented in Table 2.4. Therefore it is theoretically possible to select a combination of connexins that displays suitable experimental behavior. For example, connexins 26 and 43 do not interact, and are known to form distinct and separate plaques when co–expressed in the same cell. Furthermore, reverse gating behavior has been observed in proline mutants of Cx26 and Cx32 mutants where the mutation occurs in the M1 domain of the protein. These reverse gaters are closed in a resting state, and open upon the application of current, this behavior being the exact opposite of that displayed by wild–type connexins. In our thought experiment, the select cell C_S would express both Cx43 and Cx26. Cell C_A would express Cx45, which interacts with Cx43 but not 26, while cell C_B would express a reverse–gating mutant of Cx26. These conditions should satisfy the parameters outlined initially. Based on observed heterotypic interactions, ocnnexins can be classified into two groups that preferentially interact among themselves, as in Table 3. Such behavior can be used to generate a combinatorial connexin library as mentioned earlier.

However this design of a 1–bit multiplexer exhibits some limitations. One problem is that the C_S also acts as the output cell, and hence the output has to be measured in C_S. An extra cell could be attached for measuring output, but would make the experiment complicated. Also, when the state is changed more than once, both the dyes from the input cells will be present in C_S, and one needs a mechanism to eliminate this interference.

4 Conclusion

Hence, we have discussed various properties of gap junctions that make them suitable as basic building blocks for circuits and for use in nanotechnology. How-

ever many practical and technical issues need to be resolved before experimental realization of the aforementioned ideas is possible. Nonetheless the potential application of gap junctions in these areas remains a novel and interesting idea.

References

[1] Simon A.M. and Goodenough D.A. Diverse functions of vertebrate gap junctions. *Trends in Cell biology*, 1998.

[2] Scemes E., Suadicani S.O., and Spray D.C. *Gap Junctions: Molecular Basis of Cell Communication in Health and Disease*, volume 49 of *Current Topics in Membranes*, chapter 7: Intercellular calcium wave communication via gap junction dependent and independent mechanisms, pages 145–174. Academic Press, 2000.

[3] Bukauskas F.F., Bukauskiene A., and Verselis V.K. Conductance and permeability of the residual state of connexin43 gap junction channels. *Journal of General Physiology*, 2002.

[4] Sosinsky G. *Gap Junctions: Molecular Basis of Cell Communication in Health and Disease*, volume 49 of *Current Topics in Membranes*, chapter 1: Gap Junction Structure: New Structures and New Insights, pages 2–23. Academic Press, 2000.

[5] Abelson H., Allen D., Coore D., Hanson C., Homsy G., Knight T.F., Nagpal R., Rauch E., Sussman G.J., and Weiss R. Amorphous computing. Technical Report AI Memo No. 1665, MIT Artificial Intelligence Laboratory, 1999.

[6] Skerrett I.M., Smith J.F., and Nicholson B.J. *Gap Junctions: Molecular Basis of Cell Communication in Health and Disease*, chapter 12: Mechanistic differences between mechanical and eletrical gating of gap junctions. Academic Press, 2000.

[7] Harris Andrew L. Emerging issues of connexin channels: biophysics fills the gap. *Quarterly Reviews of Biophysics*, 2001.

[8] Millonas Mark M. and Rauch Erik M. Trans–membrane signal transduction and biochemical turing pattern formation.

[9] Falk M.M., Buehler L.K., Kumar N.M, and Gilula N.B. Cell–free synthesis and assembly of connexins into functional gap junction and membrane channels. *The EMBO Journal*, 1997.

[10] Kumar N.K and Gilula N.B. The gap junction communication channel. *Cell*, 1996.

[11] Phelan P. *Gap Junctions: Molecular Basis of Cell Communication in Health and Disease*, volume 49 of *Current Topics in Membranes*, chapter 19: Gap Junction Communication in Invertebrates: The Innexin Gene Family, pages 390–422. Academic Press, 2000.

[12] Bruzzone R., White T.W., and Goodenough D.A. The cellular internet: on–line with connexins. *BioEssays*, 1996.

[13] Rozental R., Giaume C., and Spray D.C. Gap junctions in the nervous system. *Brain Research Reviews*, 2000.

[14] Weiss Ron and Knight Thomas F. Engineered communications for microbial robotics. In *DNA6: Sixth International Meeting on DNA Based Computers*.

[15] Ahmad S., Diez J.A., George C.H., and Evans W.H. Synthesis and assembly of connexins *in vitro* into homomeric and heteromeric functional gap junction hemichannels. *Biochemical Journal*, 1999.

[16] Hofer T., Venance L., and Giaume C. Control of plasticity of intercellular calcium waves in astrocytes: a modeling approach. *Journal of Neuroscience*, 2002.

[17] Suchyna T.M., Nitsche J.M, Chilton M., Harris A.L., Veenstra R.D., and Nicholson B.J. Different ionic selectivities for connexins 26 and 32 produce rectifying gap junction channels. *Biophysical Journal*, 1999.

[18] Unger V.M., Kumar N.M., Gilula N.B., and Yeager M. Three–dimensional structure of a recombinant gap junction membrane channel. *Science*, 1999.

INDEX TO STUDENTS