DNA-BASED SELF-ASSEMBLY AND NANOROBOTICS: THEORY AND EXPERIMENTS

by

Sudheer Sahu

Department of Computer Science
Duke University

Date: _____

Approved:

_____
John Reif, Supervisor

_____
John Board

_____
Alexander Hartemink

_____
Thomas LaBean

_____
Kamesh Munagala

_____
Xiaobai Sun

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Computer Science
in the Graduate School of
Duke University

2007

ABSTRACT

DNA-BASED SELF-ASSEMBLY AND NANOROBOTICS: THEORY AND EXPERIMENTS

by

Sudheer Sahu

Department of Computer Science
Duke University

Date: _____

Approved:

_____
John Reif, Supervisor

_____
John Board

_____
Alexander Hartemink

_____
Thomas LaBean

_____
Kamesh Munagala

_____
Xiaobai Sun

An abstract of a dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Computer Science
in the Graduate School of
Duke University

2007

Abstract

We study the following fundamental questions in DNA-based self-assembly and nanorobotics: How to control errors in self-assembly? How to construct complex nanoscale objects in simpler ways? How to transport nanoscale objects in programmable manner?

Fault tolerance in self-assembly: Fault tolerant self-assembly is important for nanofabrication and nanocomputing applications. It is desirable to design *compact* error-resilient schemes that do not result in the increase in the original size of the assemblies. We present a comprehensive theory of compact error-resilient schemes for algorithmic self-assembly in two and three dimensions, and discuss the limitations and capabilities of redundancy based compact error correction schemes.

New and powerful self-assembly model: We develop a reversible self-assembly model in which the glue strength between two juxtaposed tiles is a function of the time they have been in neighboring positions. Under our time-dependent glue model, we can rigorously study and demonstrate catalysis and self-replication in the tile assembly. We can assemble thin rectangles of size $k \times N$ using $O(\frac{\log N}{\log \log N})$ types of tiles in our model.

Modeling DNA-based Nanorobotical Devices: We present a framework for a discrete event simulator for DNA-based nanorobotical systems. It has two major components: a physical model and a kinetic model. The physical model captures the conformational changes in molecules, molecular motions and molecular collisions. The kinetic model governs the modeling of various reactions in a DNA nanorobotical system such as hybridization, dehybridization and strand displacement.

DNA-based molecular devices using DNAzyme: We design a class of nanodevices that are autonomous, programmable, and require no protein enzymes. Our DNAzyme based designs include (1) *DNAzyme FSA*, a finite state automata device , (2) *DNAzyme router* for programmable routing of nanostructures on two-dimensional DNA addressable lattice, and

(3) *DNAzyme doctor*, a medical-related application that respond to the under-expression or over-expression of various RNAs, by releasing an RNA.

Nanomotor Powered by Polymerase: We, for the first time, attempt to harness the mechanical energy of a polymerase $\phi 29$ to construct a polymerase based nanomotor that pushes a cargo on a DNA track. Polymerase based nanomotor has advantage of high speeds of polymerase.

Contents

List of Tables

List of Figures

Acknowledgements

To begin with, I would like to express my sincere thanks to my advisor, Prof. John Reif, for his constant support and encouragement throughout my PhD years. His persistent motivation, farsighted guidance, extensive knowledge and endless pool of ideas were always a source of inspiration and helped me to overcome the technical difficulties along the way. I could had never imagined writing this dissertation, if not for all those intellectually stimulating discussions with John in the office D223, on our way to coffee, and during travel to various conferences. His excitement, whenever we inched forward towards solving any research problem, always motivated me towards making more progress.

I would also like to express my gratitude towards Prof. Thom LaBean (from my PhD committee). He has been a tremendous help throughout my work in the area of DNA based nanotechnology. His expertise in biochemistry was always extremely useful. He often initiated excellent discussions in lab meetings and journal clubs. His critical analysis of my ideas coupled with his own novel ideas was of great use, specially in the nanomotor project.

I also wish to thank Prof. Xiaobai Sun (PhD committee member). She always provided me with valuable suggestions related to research and career. She helped me a lot by providing useful feedback on my dissertation, and presentation of my work.

I am also thankful to Prof. Alexander Hartemink (PhD committee member). Two of his courses, in my first year at Duke, were my first encounter with interdisciplinary area of biology and computation. Further explorations, introduced me to the exciting area of DNA computing, in which I chose to work for my PhD. He has been instrumental in giving me tips on writing the dissertation. I have benefitted greatly from his feedbacks and advices time to time.

I would also like to thank Prof. John Board (PhD committee member) for his valuable feedbacks on my research and dissertation during my PhD years.

I am also grateful to Prof. Kamesh Munagala (PhD committee member), who was always very approachable, and I always got useful suggestions on research and career from him.

I would like to extend my thanks to other present and past members of our DNA nanotechnology group: Hanying Li, Josh Carter, Urmi Majumder, Nikhil Gopalkrishnan, Geetha Shetty, SungHa Park, and Peng Yin for excellent discussions during lab meetings and journal clubs, and for answering my wide range of queries inside and outside lab. In particular, I got an opportunity to collaborate with Peng Yin on a lot of projects during my earlier years at Duke that helped me get started with research early. I would also like to thank Bei Wang, Sam Slee, and Piyush Shivam from Computer science Department for helpful discussions.

I would also like to thank Graduate Secretary of Computer Science Department, Diane Riggs, who takes extremely good care of all the graduate students in the Department, and hence makes our lives much easier.

Big pillars of support that I am lucky to have, are my friends *in Duke/Durham*: Vijeta, Kesari, Amit Singh, Nitin Goel, Rakesh, Pradeep, Dhiraj, etc; and *outside Durham*: Jay, Nikhil, Vikas Varshney, Saurabh Shrivastav, etc. They always helped me bounce back whenever I was down and out.

In the end, I would like to express my gratitude to my family: my parents and my siblings, and my fiancee. My parents always had full confidence in me, never put any sort of pressure on me, and always let me decide independently on every matter. I am grateful to them for the values, that they taught me very early in life and that will stay with me forever. My mother gave me the very first lessons in mathematics by teaching me how to count. My father, a mathematics teacher, instilled in me a love for the subject from the very beginning. My younger brother Vidosh (Monu) and sister Nidhi (Appu) gave me a perfect company so that I had a really funfilled childhood. Last but not the least I express my thanks to my fiancee Sandeepa for being in my life, adding some spice to it, and making it much more colorful. She has demonstrated exemplary patience in being a good listener to all my blabberings.

Thanks everyone for being there.

Chapter 1

Introduction

Self-assembly is a ubiquitous and autonomous process in which small objects combine together to form larger and complex structures. Examples in nature are numerous: atoms self-assemble into molecules, cells into tissues, tissues into organs, and so on. Recently, it has been demonstrated as an efficient mechanism for bottom-up construction of nanostructures in nanotechnology [158, 195, 109, 94, 202, 201, 41, 104]. The potential of self-assembly is not limited to nanofabrication. The ability of two-dimensional and three-dimensional assemblies to perform parallel universal computations has been explored in development of self-assembly of DNA tiles as a tool for nanocomputation [96, 132, 189, 196, 200]. The potential advantage of significantly high circuit density in molecular circuits as compared to the traditional microelectronic circuits, is chief motivation for developments of molecular circuit components [128, 16, 44, 131, 212]. Molecular scale circuits need the bottom-up approach of self-assembly of DNA tiles to get assembled out of these molecular electronic components. For the selective attachment of the molecular electronic components to particular tiles of DNA tiling, [185] prepared molecular DNA-linked systems, while [35] used directed self-assembly of molecular terrace structures in organic monolayers. Self-assembly has been demonstrated at larger scales (meso-scale) using capillary forces for interactions between meso-scale tiles [32, 146].

With the ease in construction of complex nanostructures becoming evident, researchers focused on designing controllable robotic devices that can move on these nanostructures and perform specific tasks. DNA-based molecular devices have the advantage of being relatively simple to design and engineer, due to the predictable secondary structure of DNA nanostructures and the well-established biochemistry used to manipulate DNA

nanostructures. A variety of DNA nanomechanical devices mediated by external environment [10, 58, 102, 162, 163, 176, 203, 210, 110, 160, 161] that exhibit motions such as open/close[162, 163, 210], extension/contraction [10, 58, 102], and rotation [110, 176, 203], have been demonstrated. Recent times have seen significant progress in construction of DNA nanomechanical devices that execute autonomous, progressive motions[134, 135, 178, 208].

Even though self-assembly is a fundamental natural phenomenon, it is not very well understood, and hence a small fraction of its immense potential is being currently used. The errors that creep in the self-assembly are posing the biggest hurdle in letting it make a significant impact on nanotechnology. The construction of complex nanostructures by self-assembly of current DNA tiles is extremely complicated and demanding. The potential application of DNA nanorobots performing useful transportation on nanostructures also faces tough challenges ahead because of lack of programmability and speed in current DNA-based nanodevices.

This thesis is an attempt at addressing these challenges in DNA-based self-assembly and nanorobotics, and in particular study the following fundamental questions:

- How to control errors in self-assembly ?
- How to construct complex nanoscale objects in simpler ways?
- How to transport nanoscale objects in a programmable manner?

I approach these questions (1) by studying the nature of the self-assembly process analytically, in silico and experimentally, and (2) by designing, analyzing and implementing novel nanorobotical systems.

1.1 Self-Assembly of DNA Tiles

In DNA-based self-assembly our focus is on the following two aspects:

- study of mathematical models of self-assembly, and design of advanced, novel and powerful self-assembly models that help in realizing the vast potential of self-assembly.
- error correction in two and three dimensional self-assembly.

1.1.1 Mathematical Models of Self-Assembly

Mathematical models of self-assembly deal with self-assembly of DNA tiles. However, the assembly of tiles is not a new mathematical problem. It has been well studied in 60s and 70s, though in a more abstract context.

History of Tiling Problem

In 1961, Wang [183] defined a class of tiling problems as follows. We are given a finite set of square tiles of unit size each with top and bottom sides labeled with symbols from a finite alphabet as shown in Figure 1.1.

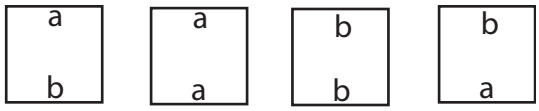


Figure 1.1: Four different kind of tiles. Each tile has two symbols-one on top side and one on bottom side

We are also given the initial placement of a subset of these tiles, and the borders of the region where tiles must be placed defining the extent of tiling. The problem is to place the tiles, chosen with replacement, in all these square regions within the specified borders, so that each pair of vertical abutting tiles have identical symbols on their contacting sides as shown in Figure 1.2. These problems are also known as domino tiling problem.

In 1962, Buchi [34] proved that given a finite set of tile types, the domino tiling problem is undecidable if the extent of tiling is the positive quadrant of the plane and a single tile is required to be at a fixed location. In 1966, Berger [23] gave a proof that removed the latter condition, thus proving the undecidability of the tiling problem. In 1971, proof of

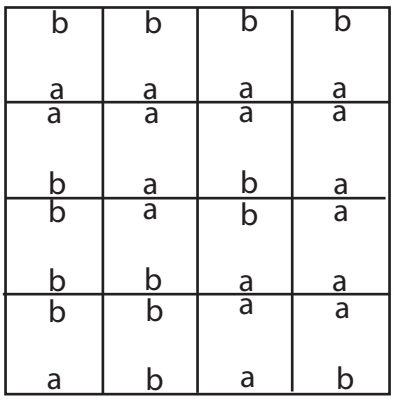


Figure 1.2: An arrangement of tiles to fill a square region

Robinson [141] provided a direct simulation of a single tape deterministic Turing Machine to prove this undecidability result.

In 1977, Garey, Johnson and Papadimitriou [64] proved that the domino tiling problem is NP-complete if the extent of tiling is a rectangle of polynomial size. In 1981, Lewis and Papadimitriou [101] gave a direct proof of this NP-completeness result, providing a simulation of a single tape nondeterministic Turing Machine running in time $T \geq n$ and space $S \leq n$ by assembly of an $S/2 \times T$ array of tiles.

In 1995, Winfree [188] proposed the idea of using DNA cross-over molecules as tiles, with sticky ends serving as the sides of the tiles. This revolutionized the whole field of DNA computing, because the concept of computation by self-assembly of DNA tiles came into existence with this. Figure 1.3 shows the picture of a DNA double crossover molecule (DX tile) that can be used as a tile.

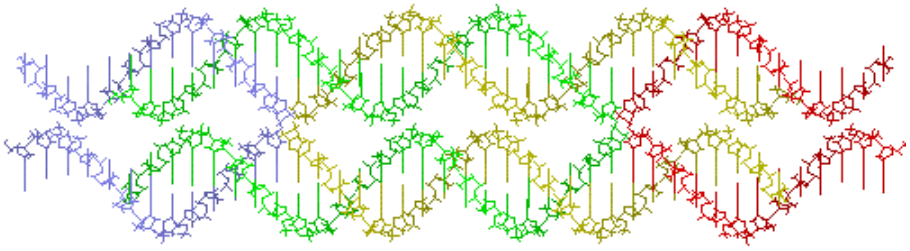


Figure 1.3: A double cross-over molecule with four sticky ends

Figure 1.3 shows one DX tile and its four sticky ends. Each DNA tile shown in Figure 1.3 can assemble with four neighbor DNA tiles with each of its sticky ends attached to one DNA tile.

It was the advent of self-assembly of DNA tiles, which motivated people to study the problem of assembly of tiles in a rigorous way. Various models were proposed to model this process. There is one irreversible model for the process (given by Winfree and Rothemund [194]) and one reversible model for the process (given by Adleman [4, 7]). We will discuss both of them next.

Winfree and Rothemund's Abstract Tile Assembly Model

A tile over Σ is a unit square where each side is colored from the set Σ of glues; formally, a tile is a 4-tuple $(\sigma_N, \sigma_E, \sigma_S, \sigma_W) \in \Sigma^4$ indicating the glues on the north, east, south and west sides of the tile. A function $g : \Sigma \times \Sigma \to R$, is a strength function. They considered g such that mismatched sides have no interaction strength and matching sides have positive strengths.

$$g(\sigma, \sigma') = \begin{cases} +ve & \text{if } \sigma = \sigma' \\ 0 & \text{otherwise} \end{cases}$$

A tile may be added to an assembly if the summed strength of its interactions with its neighbors exceeds a threshold, called temperature. Tile complexity of a shape is the minimum number of distinct non-empty tiles required to uniquely produce that shape. Adleman [5] proved the tile complexity of a $N \times N$ square is $\Theta(\frac{\log N}{\log \log N})$.

Adleman's Reversible Model

Adleman proposed a reversible model for the study of self-assembly [4, 7]. In this model a tile attached to the assembly may detach from it. Adding reversibility allows the theory to more accurately model real physical systems and brings thermodynamics to bear on self-

assembly problems. Adleman studied equilibrium behaviour of linear polymerization of tiles in this model and proved that starting from any initial conditions the system reaches very near the equilibrium if the number of time-steps are more than a certain value.

Winfree's Kinetic Assembly Model

Winfree proposed DNA double-crossover molecules to self-assemble to form an algorithmically patterned two-dimensional lattice and developed a realistic model of DNA self-assembly based on the thermodynamics and kinetics of oligonucleotide hybridization. He made the following assumptions to simply the model:

1. Monomer concentrations are held constant. In addition, all monomer types are held at the same concentration.

2. Aggregates do not interact with each other; thus the only reaction to model are the addition of a monomer to an aggregate, and the dissociation of a monomer from an aggregate.

3. As in the hybridization of oligonucleotides, the forward rate constants for all monomers are identical.

4. As in the hybridization of oligonucleotides, the reverse rate depends exponentially on the number of base-pair bonds which must be broken, and mismatched sticky ends make no base-pair bonds.

Generalized Models

Apart from these basic models, various generalized models of self-assembly are also studied [9]: namely, multiple temperature model, flexible glue model, and q-tile model. It was proved that the tile complexity of $N \times N$ squares can be reduced to $\Theta(\sqrt{\log N})$.

While $\Omega(\frac{N^{1/k}}{k})$ types of tiles are required for the self-assembly of a thin rectangle of size $k \times N(k < N)$, it can be assembled by $O(\frac{\log N}{\log \log N})$ type of tiles.

Though all these models contribute greatly towards a good understanding of the process of self-assembly, there are still a few things that could not be easily explained or modeled (for example, the process of catalysis and self-replication in tile assembly). In Chapter 3, we propose a new model, in which these processes can be studied. In this new model, which is built on the basic framework of above mentioned models, the glue strength between different glues is dependent on the time for which they have remained together.

1.1.2 Error Correction in Self-Assembly

In DNA assemblies, incorrect tiles are incorporated in the growing structure with error rates ranging from 1% to 5%[193]. There are two approaches to combat the errors. The first is to reduce the inherent error rate by optimizing the physical conditions [190] or using newer molecular mechanisms [42], while the other approach is to improve the tile design so that the total number of errors in the final structure is reduced in spite of the intrinsic error-rate remaining the same[43, 137, 193].

Winfree [193] laid the foundations of work towards improving the tile-design to reduce the errors in assembly. It required replacing a tile by a larger block of tiles. Though it resulted in scale up of the total size of assembly, to be 4 times for error reduction to ϵ^2 and 9 times for error reduction to ϵ^3, it paved the way for further work in error-reduction. Later, the snaked proof-reading scheme that could correct both growth and nucleation errors in the self-assembly was built upon this construction [43]. Later a method was proposed to control nucleation errors programmably [155].

However, each of these schemes significantly scaled up the overall size of assembly by replacing each tile in original assembly with multiple tiles. In applications like molecular fabrication tasks where the scale of final pattern is of critical importance, this scaling up is

undesirable. Reif et al.[137] proposed a *compact error-resilient tiling schemes* in which errors could theoretically be reduced to ϵ^2 (2-way overlay redundancy) and ϵ^3 (3-way overlay redundancy) without increasing the size of the assembly. Recall that in nanocomputation using algorithmic self-assembly, output values in a row act as input values for the next row. A distinction of this scheme [137] was that unlike other methods, it considered the error resilience in the whole assembly and not just in the final output row. This is important in the assembly of a nanostructure of desired pattern, where any incorrect placement of any tile is a defect (even though it might not have interfered with the subsequent growth of assembly). But it had its limitations on the Boolean functions that could be used for the error-resilient algorithmic assembly. In particular, it required one of the function to be *XOR*, and for reduction to ϵ^3 the additional requirement was that the other function should be input-sensitive to one of the inputs. A Boolean function $f(x)$ is called *input-sensitive* to a Boolean variable x if whenever x changes $f(x)$ also changes. It is thus a critical challenge to improve these compact error-correction schemes to incorporate any arbitrary Boolean functions. In case that is not possible, it is important to characterize the class of Boolean functions to which these error-correction schemes can be extended.

Recently Winfree[168] presented a *compact error resilient scheme* based on Chen et al [43]. They were also concerned by only the errors that affected the final output of the nanocomputation by algorithmic self-assembly.

The Challenge of 3D Tiling Assembly Error-Correction

Self-assembly in three dimensions is extremely promising in the field of microelectronics assembly, where independent manipulation of each component is required. It is already being seen as promising candidate for heterogeneous three-dimensional integration of next-generation microsystems[49, 113, 187, 198]. Apart from this, the potential advantages of three-dimensional structures over two-dimensional structures in nanofabrication includes

a considerably increased circuit density. Jonoska et al [82] proposed the use of three-dimensional DNA structures in computing. Simple examples of algorithmic computation in three dimensions includes the generalization of Pascal triangle to 3D[29] and three dimensional multiplexers (the latter would provide a mechanism for 3D memory addressing with the appropriate affixed molecular electronic components). Recently crystal structure of three-dimensional DNA lattices formed by self-assembly was demonstrated [126]. The question of fault-tolerance naturally arises with the increasing popularity of self-assembly for construction of three dimensional self-assembled structures. It will be critical to determine how successfully can the error-correction techniques used for two-dimensional assemblies be extended to three-dimensions.

The Challenge of Self-Healing Tiling Assemblies

The one property of biological systems that makes them robust is their ability to self-heal in case of damage. *Self-healing* is essentially the self-assembly of the constituent elements in the damaged part of a system, so as to repair the damage. Damage to living cells can be caused by an external intruder or some mechanical impulse or unfavorable physical conditions. It is an interesting and important challenge to design DNA tiles that form lattices having the ability to self-heal, thereby imparting them the much desired robustness to withstand environmental damage. Winfree[192] gave a construction for self-healing in a two-dimensional assembly in which he replaced a single tile with 3×3 (for simple assemblies like Sierpinski triangles), 5×5 (for general assemblies) and 7×7 (for additional robustness to nucleation errors) blocks of tiles.

In Chapter 2, we present a comprehensive theory of compact error resilient schemes. First we present a compact error correction scheme in two dimensional self-assembly that reduces the error from ϵ to ϵ^2 for arbitrary Boolean functions. Then we characterize the class of Boolean functions for which error reduction from ϵ to ϵ^3 is possible using redun-

dancy based compact error resilient schemes. Also we prove that error reduction from ϵ to ϵ^4 is impossible using redundancy based compact error resilient schemes. Next we examine three-dimensional self-assembly. First we present a compact error resilient scheme that reduces error to ϵ^2 for arbitrary Boolean functions and ϵ^3 for a restricted class of input-sensitive Boolean functions. We also prove that error reduction to ϵ^4 can not be obtained for arbitrary Boolean functions using redundancy based compact error resilient schemes. We also extend the idea of Winfree's construction for self-healing in two-dimensions [192] to three-dimensional assembly.

1.2 DNA-based Nanorobotics

In DNA-based nanorobotics, our focus is on the following two aspects:

- Design and implementation of novel DNA-based nanomechanical devices that are programmable, autonomous, reliable and fast.
- Modeling nanorobotical devices accurately in order to support their design process.

1.2.1 DNA Nanomechanical Devices

Prior Nonautonoumous Nanomechanical DNA Devices

Recent years have seen an ever-increasing interest of researchers in designing DNA-based nanomechanical devices. A variety of DNA nanomechanical devices have been constructed that exhibit motions such as open/close [162, 163, 210], extension/contraction [10, 58, 102], and rotation [110, 176, 203]. The motion of these devices is mediated by external environmental changes such as the addition and removal of DNA fuel strands [10, 58, 102, 162, 163, 176, 203, 210] or the change of ionic strength of the solution [110]. For example, non-autonomous progressive walking devices, mediated by the addition and removal of DNA strands, were constructed both by Seeman [160] and Pierce [161]. Although

in many cases ingeniously designed, these devices need external (human or automation-based) intervention for each step of their motions. These synthetic DNA devices are in sharp contrast with cellular protein motors and machines on macroscale that operate autonomously, without requiring any external interference.

Prior Autonoumous DNA Nanomechanical Devices

There has been a lot of focus on designing autonomous DNA devices that can act without human assisted lab procedures. Though exciting progress has been made in building autonomous DNA computational devices, those devices are limited by their incapability to preserve the input data for subsequent processing after the first round of computation-the input is either destroyed [22, 20] or blocked [108, 94]. Recent times have seen significant progress in construction of DNA nanomechanical devices that execute autonomous, progressive motions. Designs of such nature were first envisioned by Reif [134, 135]. However the potential power of these devices is limited by the fact that they undergo only random bi-directional movements. Turberfield et al proposed using DNA hybridization energy to fuel autonomous free-running DNA machines [178]. Yin et al [208] was the first to experimentally demonstrate an autonomous DNA walker, which is an autonomous DNA device in which a DNA fragment translocates unidirectionally along a DNA nanostructure. It makes use of alternating actions of restriction enzymes and ligase. The action of ligase is powered by ATP consumption.

DNAzyme Based Nanomechanical devices

Recently Mao demonstrated two autonomous DNA nanomechanical devices driven by DNA enzymes (non-protein), namely (a) a tweezer [46, 45] which is a DNA nanostructure that open and closes autonomously and (b) a DNA crawler [175] using DNA enzyme, which traverses across a DNA nanostructure.

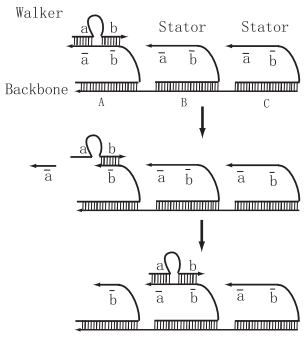


Figure 1.4: Overview of Mao's crawler [175] constructed using DNA enzyme

Their crawler device contains a DNA enzyme (DNAzyme) that constantly extracts chemical energy from its substrate molecules (RNA) and uses this energy to fuel the motion of the DNA device. This DNAzyme based crawler integrates DNAzyme activity and strand-displacement reaction. They use 10-23 DNAzyme, which is a DNA molecule that can cleave RNA with sequence specificity. The 10-23 DNAzyme contains a catalytic core and two recognition arms that can bind to a RNA substrate. When the RNA substrate is cleaved, the short fragment dissociates from the DNAzyme and that provides a toehold for another RNA substrate to pair with a short recognition arm of the DNAzyme. The crawler device traverses on a series of RNA stators implanted on a nanostructure as shown in Figure 1.4. While an ingenious device, there are a number of limitations of Mao's DNAzyme based crawler: (1) it did not demonstrate the loading and unloading of nanoparticles (2) it only traverses along a one dimensional sequence of ssRNA strands (stators) dangling from a DNA nanostructure, and its route is not programmable (3) it does not execute finite state transitions beyond what are required to move (that is, it does not execute computations).

In Chapter 5, we attempt to address the above limitations. We present a class of DNAzyme based nanodevices with substantially enhanced functionalities to the prior DNAzyme based crawler previously developed. Their crawler is the primary inspiration to our designs. All the devices described in this chapter are based on selective cleaving activity of DNAzyme and strand displacement processes.

We have known polymerase as an enzyme that copies DNA or RNA template and thus forms the basis of the life-processes. Figure 1.5 illustrates the way a polymerase extends a primer. In Chapter 6, we present a nanomotor that exploits the energy of polymerase

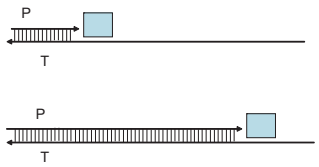


Figure 1.5: A schematic showing a template, a primer, and a polymerase (box) that extends the primer using the nucleotides available in the solution

to push cargo on a DNA track. The idea is innovative and opens a completely new frontier in DNA-based nanotechnology. We use the polymerase $\phi 29$ for its exceptional strand displacement capabilities. The inherent advantage of using a nanomotor driven by polymerase against any other existing DNA nanomechanical device is its fast speed. A typical $\phi 29$ polymerase can travel at the rate of about 2000 nucleotides per minute [2] at room temperature, that translates to approximately 680 nanometers per minute on a nanostructure.

1.2.2 Nanorobotic Simulators

Major challenges in front of researchers interested in designing complex DNA-based nanodevices, are the time consuming and costly experiments. A lot of times the effect of alterations in only a few parameters need to be tested, and the entire set of experiments need to be repeated from scratch. Accurate computer simulations that capture the essential physical and chemical properties can serve as an effective tool in the design process.

Prior Simulators for DNA Computing

Previous simulators for DNA computing include:

- *VNA simulator*[121, 122]: a simulator to aid the protocols for DNA computing. The simulator consists of two main parts, one for finding reactions among existing molecules and generating new ones, and the other for numerically solving differential equations to calculate the concentration of each molecule.

- *Virtual test tubes*[65, 66, 67]: a simulator for biochemical reactions based on the kinetics of molecular interactions.

- *Hybrisim*[77]: a simulator that deals with the detailed simulation of hybridization only between two strands, and therefore, very limited in use.

Bois et. al. [27] investigated the possible effects of topological constraints in DNA hybridization kinetics. Recently Dirks et. al. [55] developed an algorithm aimed at analyzing the thermodynamics of unpseudo-knotted multiple interacting DNA strands in a dilute solution. None of these deal with the shapes of nanostructure, and therefore not suitable for simulation of nanorobotics or nanofabrication applications.

In Chapter 4, we describe a comprehensive framework for building a modeling tool for DNA-based nanorobotic devices. We also provide a preliminary implementation to show the feasibility of the approach. It is explicitly divided into two parts: physical modeling and kinetic modeling. Physical modeling deals with the physical motion of the molecules, and their conformations. Kinetic modeling part controls the event simulation based on thermodynamic properties of the molecules.

1.3 Contribution

We study the following aspects of DNA-based nanotechnology: *fault tolerance in self-assembly*, *new and powerful self-assembly models*, *modeling DNA-based nanorobotical devices*, and *design and implementation of novel nanodevices*. Fault-tolerance in self-assembly is one of the most important challenge in self-assembly, as the errors in self-

assembly are a major hurdle in the areas of nanocomputing and nanofabrication. The development of newer models of self-assembly helps to construct complex nanostructures in simpler ways. In the area of nanorobotics, a good modeling tool for nanorobotical devices, can provide immensely useful foresight during design. A framework for building such a modeling tool provides the basic foundation needed. The novel designs of autonomous and programmable DNAzyme based devices pave way for design of more sophisticated DNA-based nanodevices. The experimental demonstration of efficient fast moving polymerase based nanomotor opens new frontiers in DNA-based nanotransportation devices.

Compact Error Resilient Schemes in Self-Assembly: Fault tolerance is definitely the highest priority requirement for self-assembly to have considerable impact in the areas of nanofabrication and nanocomputing. It is desirable to design *compact* error-resilient schemes that do not result in the increase in the original size of the assemblies. In Chapter 2, we present an exhaustive theory of compact error-resilient schemes for algorithmic self-assembly in two and three dimensions, in which we discuss the limits and capabilities of redundancy based compact error correction schemes. Further, we develop the first provable compact error resilience schemes for three dimensional tiling self-assemblies. We also extend the work of Winfree on self-healing in two-dimensional self-assembly to obtain a self-healing tile set for three-dimensional self-assembly.

Self-Assembly Model of Time Dependent Glue Strength: The development of new and powerful self-assembly models can assist in realization of complex nanostructures and phenomena with relative ease. In Chapter 3, we develop a reversible self-assembly model in which the glue strength between two juxtaposed tiles is a function of the time they have been in neighboring positions. It can be implemented using strand displacement reactions on DNA tiles. Under our model, we can for the first time demonstrate and study catalysis and self-replication in the tile assembly rigorously. We can assemble thin rectangles of size $k \times N$ using $O(\frac{\log N}{\log \log N})$ types of tiles, which is a significantly lower than the lower bound

in *Tile Assembly Model*.

Framework for Modeling DNA-based Nanorobotical Devices Recent successes in building large scale DNA nanostructures and in constructing DNA nanomechanical devices have inspired scientists to design more complex nanoscale systems. The design process can be made considerably more efficient and robust with the help of simulators that can model such systems accurately prior to their experimental implementation. In Chapter 4, we design a framework for a discrete event simulator for simulating the DNA-based nanorobotical systems. It has two major components: a physical model and a kinetic model. The physical model captures the conformational changes of molecules, molecular motions and molecular collisions. The kinetic model governs the modeling of various chemical reactions in a DNA nanorobotical systems including the hybridization, dehybridization and strand displacement.

DNAzyme based Autonomous Nanodevices: A major challenge in nanoscience is the design of synthetic molecular devices that run *autonomously* and are *programmable*. In Chapter 5, we present the design of a class of DNA-based molecular devices using DNAzyme. These DNAzyme based devices are autonomous, programmable, and further require no protein enzymes. Our DNAzyme based designs include (1) a finite state automata device, *DNAzyme FSA* that executes finite state transitions using DNAzymes, (2) extensions to it including probabilistic automata and non-deterministic automata, (3) its application as a *DNAzyme router* for programmable routing of nanostructures on a 2D DNA addressable lattice, and (4) a medical-related application, *DNAzyme doctor* that provide transduction of nucleic acid expression: it can be programmed to respond to the under-expression or over-expression of various strands of RNA, with a response by release of an RNA.

Nanomotor Powered by Polymerase: Polymerase has long since been known as responsible for sustenance of life forms in the world, by polymerization of new DNA or RNA against an existing DNA or RNA template in the processes of replication and tran-

scription. In Chapter 6, we, for the first time, attempt to harness the mechanical energy of a polymerase enzyme $\phi 29$ in order to push a cargo, and hence construct a nanomotor from a polymerase. This nanomotor has inherent superiority to other existing nanomotors because of the speed of polymerase $\phi 29$ is known for its exceptional strand displacement abilities that form the basis of our nanomotor. The experimental demonstration of efficient fast moving polymerase based nanomotor open completely new frontiers in DNA-based nanotechnology.

We conclude with providing a glimpse of the potential impact of the abovementioned work in the DNA-based nanotechnology and future vision in each of the abovementioned subareas in Chapter 7.

Chapter 2

Capabilities and Limits of Compact Error Resilience Methods for Algorithmic Self-Assembly

Winfree's pioneering work laid the foundations in the area of error-reduction in algorithmic self-assembly[193]. Reif et. al. [137] contributed further in this area with compact error-resilient schemes that maintained the original size of the assemblies. It remains a critical challenge to improve these compact error resilient schemes to incorporate arbitrary Boolean functions in the algorithmic self-assembly, and to determine how far these prior results can be extended under different degrees of restrictions on the Boolean functions.

In this work we present a comprehensive theory of compact error-resilient schemes for algorithmic self-assembly in two and three dimensions. In our error model,ϵ is defined to be the probability that there is a mismatch between the neighboring sides of two juxtaposed tiles and they still stay together in the equilibrium. This probability is independent of any other match or mismatch and hence we term this probabilistic model as the *independent error model*. In our model all the error analysis is performed under the assumption of kinetic equilibrium. First we consider two-dimensional algorithmic self-assembly. We present an error correction scheme for reduction of errors from ϵ to ϵ^2 for arbitrary Boolean functions in two dimensional algorithmic self-assembly. Then we characterize the class of Boolean functions for which the error can be reduced from ϵ to ϵ^3, and present an error correction scheme that achieves this reduction. Then we prove ultimate limits on certain classes of compact error resilient schemes: in particular we show that they can not provide reduction of errors from ϵ to ϵ^4 is for any Boolean functions. Further, we develop the first provable compact error resilience schemes for three dimensional tiling self-assemblies. We

also extend the work of Winfree on self-healing in two-dimensional self-assembly[192] to obtain a self-healing tile set for three-dimensional self-assembly.

2.1 Introduction

2.1.1 Prior work in 2D Tiling Assembly Error-Correction

A major hurdle in harnessing the capabilities of algorithmic self-assembly are the errors that occur during the assembly. It has been measured that incorrect tiles are incorporated in the growing structure with error rates ranging from 1% to 5%[193, 145] in self-assembly of DNA tiles. There are two approaches to combat the errors. The first is to reduce the inherent error rate by optimizing the physical conditions [190] or using newer molecular mechanisms like strand invasion[42], while the other approach is to improve the tile design so that the total number of errors in the final structure is reduced in spite of the intrinsic error-rate remaining the same[43, 137, 193].

Winfree [193] led the work towards improving the tile-design to reduce the errors in assembly. It required replacing a tile by a larger block of tiles. Though it resulted in the total size of assembly to be 4 times for error reduction to ϵ^2 and 9 times for error reduction to ϵ^3, it paved the way for further work in error-reduction using the concept of redundancy. The basic idea was that an error in the assembly of a tile forced more errors in the immediate neighborhood of that tile, making it extremely prone to detachment, and hence reducing the error. Later, the snaked proof-reading scheme that could correct both growth and nucleation errors in the self-assembly was built upon this construction [43]. However, it required replacing a tile by a $k \times k$ block of tiles. Later a method was proposed to control nucleation errors programmably [155]. However, each of these schemes significantly scaled up the overall size of assembly. In applications like molecular fabrication tasks where the scale of final pattern is of critical importance, this scaling up is undesirable. Reif et al.[137] proposed a *compact error-resilient tiling schemes* in which errors

could be reduced to ϵ^2 (2-way overlay redundancy) and ϵ^3 (3-way overlay redundancy) without increasing the size of the assembly. The analysis of error was done in the equilibrium state of the assembly. Another distinction of this scheme was that it considered the error resilience in the whole pattern and not only in the output row. It means that this scheme had a tendency to remove any incorrectly placed tile from the assembly even if the ongoing computation was not affected by that tile. This is important in the assembly of a nanostructure of desired pattern, where any incorrect placement of any tile is a defect (even though it might not have interfered with the subsequent growth of assembly). But it had its limitations on the Boolean functions that could be used for the error-resilient algorithmic assembly. In particular, it required one of the function to be *XOR*, and for reduction to ϵ^3 the additional requirement was that the other function should be input-sensitive to one of the inputs. A Boolean function $f(x)$ is called *input-sensitive* to a Boolean variable x if whenever x changes $f(x)$ also changes. It is thus a critical challenge to improve these compact error-correction schemes to incorporate any arbitrary Boolean functions. In case that is not possible, it is important to characterize the class of Boolean functions to which these error-correction schemes can be extended. Recently Winfree[168] presented a *compact error resilient scheme* based on Chen et al [43]. They also overlooked the errors that did not affect the ongoing computation.

2.1.2 The Challenge of 3D Tiling Assembly Error-Correction

Self-assembly in three dimensions is extremely promising in the field of microelectronics assembly, where independent manipulation of each component is required. It is already being seen as promising candidate for heterogeneous three-dimensional integration of next-generation microsystems[49, 113, 187, 198]. In light of the inherent parallelism, three-dimensional nature and larger range (nanoscale to mesoscale) of application of self-assembly, it has a great potential as tool for building complex systems from microscaled

templates. Apart from this, the potential advantages of three-dimensional structures over two-dimensional structures in nanofabrication includes a considerably increased circuit density. Jonoska et al [82] proposed the use of three-dimensional DNA structures in computing. Simple examples of algorithmic computation in three dimensions includes the generalization of Pascal triangle to 3D[29] and three dimensional multiplexers (the latter would provide a mechanism for 3D memory addressing with the appropriate affixed molecular electronic components). Analogous to the simulation of a finite state automata through two-dimensional self-assembly, three dimensional self-assembly can be used to simulate a two-dimensional cellular automata, where the third spatial dimension of the 3D tiling is the time step of the cellular automata. The tiles in a horizontal plane will represent the current state of all the cells of a two-dimensional cellular automata, then the tiles assembled in horizontal plane on top of it will be states at next time instance. This allows one to derive 3D tiling assemblies from a wide variety of known two-dimensional cellular automata designs, including matrix multiplication, integer multipliers, context free language recognition, etc. Recently crystal structure of three-dimensional DNA lattices formed by self-assembly was demonstrated [126]. The question of fault-tolerance naturally arises with the increasing popularity of self-assembly for construction of three dimensional self-assembled structures. It will be critical to determine how successfully can the error-correction techniques used for two-dimensional assemblies be extended to three-dimensions.

2.1.3 The Challenge of Self-Healing Tiling Assemblies

The one property of biological systems that makes them robust is their ability to self-heal in case of damages. *Self-healing* is essentially the self-assembly of the constituent elements in the damaged part of a system, so as to repair the damage. It is a very important process in nature. The damage to the living cells can be caused by an external intruder or some mechanical impulse or unfavorable physical conditions. It is an interesting and im-

portant challenge to design the DNA tiles that form lattices having the ability to self-heal, thereby imparting them the much desired robustness to withstand environmental damage. Winfree[192] gave a construction in which he replaced a single tile with 3×3 (for simple assemblies like Sierpinski triangles) , 5×5 (for general assemblies) and 7×7 (for additional robustness to nucleation errors) block of tiles for self-healing in a two-dimensional assembly. Prior to this work, it was an open problem to find if compact self-healing tile sets could be formed and whether the techniques given by Winfree could be extended to three dimensions.

2.1.4 Our Results and Organization of this Chapter

In this chapter, we follow the notion of *compactness* as presented in [137], which requires the new error-resilient tiling assembly to be of no larger size than the original assembly. Like [137] we consider any incorrect placement of a tile anywhere in the assembly as an error and aim at reducing them as well, even though these errors might not affect the ongoing computation. As mentioned earlier, this is important for construction of nanostructures of desired pattern. In this chapter, the analysis of the error in the assembly is done in the equilibrium state of the assembly. Throughout this chapter *redundancy based compact error resilient scheme* refers to any error resilient scheme that does not scale up the size of the assembly and in which the encodings on the pads of the tiles are used to create redundancy. In the event of an error this redundancy forces more errors, which makes the incorrectly placed tiles and their neighborhoods more unstable and prone to removal from assembly, thereby reducing the error. Also we refer to k*-expansive error resilient schemes* as the error correction schemes that work by replacement of a tile by a block of multiple tiles. In case of three dimensional tiling, we carry forward this notion of *redundancy based compact error resilient schemes*.

In this chapter, we present a comprehensive theory of redundancy based compact error

resilient tiling schemes and examine the prospects of constructing compact self-healing tile sets in two and three-dimensions. The error analysis throughout this chapter is in the equilibrium state of the assembly. In Section 2.2, first we present a compact error correction schemes in two dimensional self-assembly that reduces the error from ϵ to ϵ^2 for arbitrary Boolean functions. Then we characterize the class of Boolean functions for which error reduction from ϵ to ϵ^3 is possible using redundancy based compact error resilient schemes. Also we prove that error reduction from ϵ to ϵ^4 is impossible using redundancy based compact error resilient schemes. Next in Section 2.3 we examine three-dimensional self-assembly. First we present a compact error resilient scheme that reduces error to ϵ^2 for arbitrary Boolean functions and ϵ^3 for a restricted class of input-sensitive Boolean functions. We also prove that error reduction to ϵ^4 can not be obtained for arbitrary Boolean functions using redundancy based compact error resilient schemes. In Section 2.4 we extend the idea of Winfree's construction for self-healing in two-dimensions [192] to three-dimensional assembly. In the conclusion, we review our results and state various open problems and conjectures. We conjecture stronger results that error reduction to ϵ^3 in three dimensions can not be achieved outside the previously characterized class, and error reduction to ϵ^4 is impossible to achieve for any Boolean functions using these error resilient techniques.

2.2 Error correction in Self-assembly in two dimensions

2.2.1 Assembly in two dimensions

We will consider a general assembly problem in two dimensions consisting of the assembly of a two-dimensional Boolean array of size $N \times M$, where the elements of each column are indexed from 0 to $N-1$ from right to left and rows are indexed from 0 to $M-1$ from bottom to top. The bottom row and the rightmost column provide the inputs to the assembly.

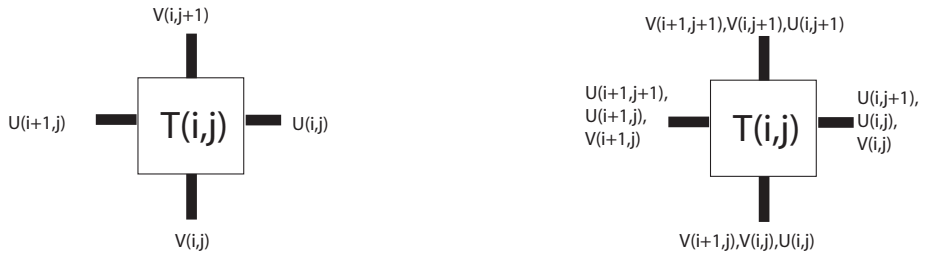


Figure 2.1: (a) Two dimensional algorithmic self-assembly (b) Construction for error reduction to ϵ^2

For $i = 0 \ldots, N-1$ and $j = 0 \ldots, M-1$:

Let $V(i,j)$ be the value of the ith column (from the right) in the jth row(from the bottom). Let $V(i,j+1)$ be the value communicated to the position $(i,j+1)$ and $U(i+1,j)$ be the value communicated to the position $(i+1,j)$. We define $U(i+1,j) = U(i,j)OP_1V(i,j)$ and $V(i,j+1) = U(i,j)OP_2V(i,j)$ for two Boolean functions OP_1 and OP_2.

Figure 2.1 a) shows a computational tile that can be used for constructing two dimensional self-assembly. Bottom and right pads are the input pads, while the pads on top and left are output pads. A pad *matches* with the neighbor's contiguous pad if the values communicated by these pads are the same. $U(i,j)$ and $V(i,j)$ are the right and bottom input pads, respectively, to the ith column from right and jth row from bottom. Then $U(i+1,j)$ the left output pad is given by $U(i+1,j) = U(i,j)OP_1V(i,j)$, while $V(i,j+1)$ the top output pad is given by $V(i,j+1) = U(i,j)OP_2V(i,j)$. The collection of tiles required for an assembly is referred to as the *tile set* for that assembly. Examples of simple two dimensional assemblies: sierpinski triangle and binary counter, and their respective tile sets, are given in [137]. Highly complex two-dimensional assemblies are possible due to the universal computability of two-dimensional self-assembly[183, 196].

2.2.2 The Error Model

We assume that error probability ϵ is defined as the probability that there is mismatch between two tiles and they still stay together in the equilibrium. This probability is independent of any other match or mismatch and hence we term this probabilistic model the *independent error model*. We also want to put emphasis on the correct assembly of all the tiles in the assembly (and hence on the correctness of complete pattern), and not just on the correctness of final output only. There might be wrong placement(s) of tile(s), that do not affect the final output of the computation. But in our error model, we count them as errors and need the error correction schemes to reduce such errors as well. In this way we differ from [168], who overlooked the errors that did not affect the ongoing computation. Consider a tile $T(i,j)$ in a $N \times M$ tiling assembly where $0 < i < N-1, 0 < j < M-1$. We define the *immediate neighborhood* of a tile $T(i,j)$ as 8 tiles surrounding it, whose coordinates differ from (i,j) by at most 1. Formally speaking, $\{T(i',j') : |i'-i| \leq 1, |j'-j| \leq 1\} \backslash \{T(i,j)\}$. Tile $T(i',j')$ is said to be *a-dependent* (for assembly dependent) on tile $T(i,j)$ if $i' \geq i$ and $j' \geq j$ and *a-independent* otherwise. Next we examine the schemes to reduce the errors in self-assembly. To reiterate, throughout this chapter, we refer to redundancy based compact error resilient scheme as error reduction scheme, where redundancy is created by encodings in the pads with absolutely no scale up of the assembly. Hence, the computation at position (i,j) is still performed at the same position. However, there is an increase in the number of type of pads for tiles.

Proposition: Under our independent error model, if an error in a pad in a tile enforces k further mismatches in the assembly in the immediate neighborhood of that tile, then error probability is reduced to ϵ^{k+1}. □

Proof. If one error guarantees k more errors, then the probability that the tile and its neighborhood in the assembly will stay together in the equilibrium in spite of these $k+1$ errors

is ϵ^{k+1}, which implies the claimed error reduction. □

2.2.3 Error reduction to ϵ^2

It is known that if an error in a tile can guarantee another error in immediate neighborhood, then it reduces the rate of errors from ϵ to ϵ^2 [193, 137]. Next we describe our construction to achieve this goal in the form of Theorem 1.

Theorem 1. *There exists a compact error correction scheme that will reduce the error from ϵ to ϵ^2 for two-dimensional algorithmic self-assembly for any arbitrary Boolean functions OP_1 and OP_2.*

Proof. **Construction** Before we begin the proof we would like to emphasize the wholeness of the pad. Each side of the tile has one pad in Figure 2.1 b), and it encodes the triplet shown in the Figure. Disagreement between corresponding elements of two such triplets in any two pads results in the total mismatch between those two pads. Consider the tile with input $U(i, j)$ and $V(i, j)$ at the right and bottom pads respectively, where $0 \leq i < N$ and $0 \leq j < M$. Our goal is to guarantee one more error in the immediate vicinity of this tile if there is one error. For that, we construct an error checking portion ($V(i, j)$) in the right side pad and one error checking portion ($U(i, j)$) in the bottom pad. We will need corresponding parts in the pads on the top ($U(i, j + 1)$) and the left side ($V(i + 1, j)$) also, which will match with the error checking parts in the bottom pad of the top neighbor $T(i, j+1)$ and right pad of the left neighbor $T(i+1, j)$ respectively. Now since top output pad depends on the value of $U(i, j + 1)$ (which is the right input of the top neighbor) we need to incorporate it in our input pads. It is necessary otherwise there will be multiple type of tiles for any given set of input pads. But for successful functioning of algorithmic self-assembly it is required that there should be only one possible tile-type for every set of input pads. So, we need one more portion in the right input pad ($U(i, j + 1)$) and hence a corresponding part in the left output pad ($U(i + 1, j + 1)$). Similarly, the need

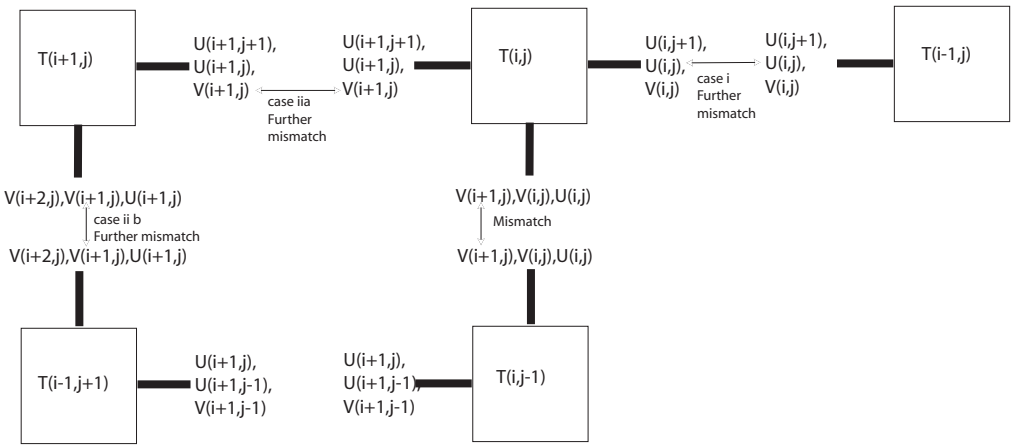


Figure 2.2: Case 1 b) A further mismatch is caused by an error in the input pads

for another portion in bottom input pad ($V(i+1,j)$) and subsequently, in top output pad ($V(i+1,j+1)$) can be explained.

This completes our description of a tile in our compact error correction scheme. It should be noted that the number of different tile types in this tile set will be 4 times as compared to number of tiles in a tile set without any error-correction. It can be attributed to the two possible values for each of $U(i,j+1)$ and $V(i+1,j)$, for every value of the inputs $U(i,j)$ and $V(i,j)$.

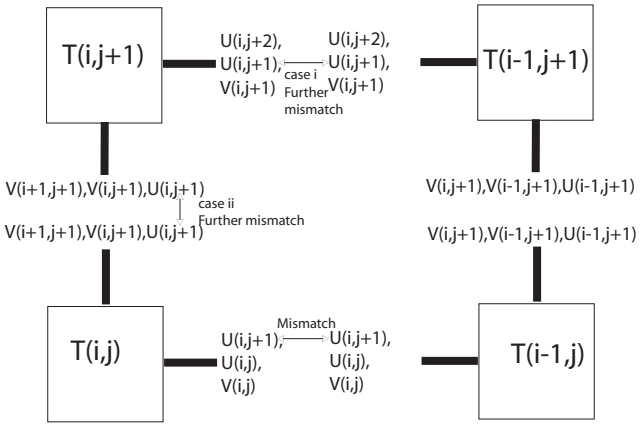


Figure 2.3: Case 2 b) of the proof

Error-Analysis: We show that if the neighborhood tiles a-independent of $T(i,j)$ are

assembled correctly then a pad binding error in any of the input pads in $T(i,j)$ causes an additional mismatch error in its neighborhood in equilibrium. We need to consider only the cases where the pad binding error occurs in either the bottom or the right pad of tile $T(i,j)$. Otherwise, if the error occurs in left (or top) pad of $T(i,j)$ then we can consider the right pad of $T(i+1,j)$ (or bottom pad of $T(i,j+1)$ for the analysis. The following case analysis provides the required proof.

1. If the bottom pad of $T(i,j)$ has a mismatch:

   (a) If $V(i,j)$ on the bottom pad has a mismatch, then $V(i,j)$ on right pad is incorrect, which causes an additional mismatch.

   (b) If $V(i,j)$ on the bottom pad is correct and $V(i+1,j)$ on bottom pad has a mismatch, $V(i+1,j)$ on left pad is incorrect (Figure 2.2). Now we will prove that it causes a further mismatch by exactly same technique as used by Reif et al[137]. We have assumed that all the rows and columns that are a-independent of tile $T(i,j)$ are correctly assembled so $T(i+1,j-1)$ is correctly assembled and has correct values of its top output pad. Hence $T(i,j)$'s left neighbor $T(i+1,j)$ is dependent upon the incorrect value communicated by the left pad of $T(i,j)$ and correct values communicated by top pad of $T(i+1,j-1)$. Now consider the pads of $T(i+1,j)$. The right pad includes $U(i+1,j+1), U(i+1,j), V(i+1,j)$ and bottom pads include $V(i+2,j), V(i+1,j), U(i+1,j)$. Since the value $V(i+1,j)$ communicated by $T(i+1,j-1)$ is correct and the value $V(i+1,j)$ communicated by $T(i,j)$ is wrong, this implies there will be a mismatch at the right or bottom pad of Tile $T(i+1,j)$.

2. If there is no error in bottom pad, but the right pad of $T(i,j)$ has mismatch:

   (a) If $U(i,j)$ on the right pad has a mismatch, then $U(i,j)$ on bottom pad is incorrect, which causes an additional mismatch.

(b) If $U(i,j)$ on right pad is correct but $U(i,j+1)$ on right pad is incorrect, then $U(i,j+1)$ on top output pad is incorrect. Now we will show that it causes a further mismatch as argued above (Figure 2.3). Since we assume that all the rows and columns that are a-independent of tile $T(i,j)$ are correctly assembled $T(i-1,j+1)$ is correctly assembled and has correct values of its left output pad. Hence $T(i,j)$'s top neighbor is dependent upon the incorrect value communicated by the top pad of $T(i,j)$ and correct values communicated by left pad of $T(i-1,j+1)$. Now consider the pads of $T(i,j+1)$. The right pad includes $U(i,j+2), U(i,j+1), V(i,j+1)$ and bottom pads include $V(i+1,j+1), V(i,j+1), U(i,j+1)$. Since $V(i+1,j)$ communicated by $T(i-1,j+1)$ is correct and the value $V(i+1,j)$ communicated by $T(i,j)$ is wrong, this implies there will be a mismatch at the right or bottom pad of Tile $T(i,j+1)$.

Hence any mismatch on the right or bottom pad of tile $T(i,j)$ causes one more mismatch in the vicinity of the tile. Together with the Proposition 2.2.2 this implies that this scheme can reduce the pad mismatch errors from ϵ to ϵ^2. □

2.2.4 Error Reduction to ϵ^3

At this point we would like to reiterate that *redundancy based compact error resilient scheme* refers to error resilient scheme that does not scale up the assembly and in which the error correction is based only on the encodings in the pads of the tiles. Also, a Boolean function $f(x)$ is said to be *input-sensitive* to Boolean input x if it changes for every change in the value of x.

Before we proceed with the error-analysis, it will be useful to understand the function class characterized in the Theorems below. Let OP_1 and OP_2 be each be two-input Boolean functions such that:

1. $U(i,j)\ OP_1\ V(i,j)$ is input-sensitive to $U(i,j)$, if $V(i,j)$ is kept constant and $U(i,j)$ $OP_2\ V(i,j)$ is input-sensitive to $V(i,j)$ if $U(i,j)$ is kept constant.

2. When both of them change at least one of the $U(i,j)\ OP_1\ V(i,j)$ or $U(i,j)\ OP_2$ $V(i,j)$ should also change.

For $U(i,j) = 0$, there are 2 possible assignments to $U(i,j)OP_1V(i,j)$ maintaining its input-sensitivity to $V(i,j)$. Similarly, for $U = 1$ there are 2 possible assignments to $U(i,j)OP_1V(i,j)$ conditioned to its input-sensitivity to $V(i,j)$. Similarly for $V(i,j)$=0 and $V(i,j)$=1 there are 2 independent assignments each. But among these half of the assignments do not satisfy the second condition. Hence the total number of Boolean functions in this class are 8. An example of such a function is given in the Table 2.1.

U	V	UOP_1V	UOP_2V
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	1

Table 2.1: An example of the OP_1 and OP_2

Define the pair of Boolean functions OP_1 and OP_2 to be *pairwise input-sensitive* if at least one of the $U(i+1,j)$ or $V(i,j+1)$ changes for any change in $U(i,j)$ or $V(i,j)$.

Theorem 2. *For arbitrary Boolean functions OP_1 and OP_2, there does not exist any redundancy based compact error resilient scheme for two-dimensional self-assembly that can reduce the error from ϵ to ϵ^3.*

Proof. For errors to reduce from ϵ to ϵ^3, an error in any input pad, say $V(i,j)$ should cause two further mismatches in the immediate neighborhood. At least one of those mismatches should be caused because of an error on one of the output pads. It should be noted that if OP_1 and OP_2 are arbitrary Boolean functions then the output $U(i+1,j)$ or $V(i,j+1)$

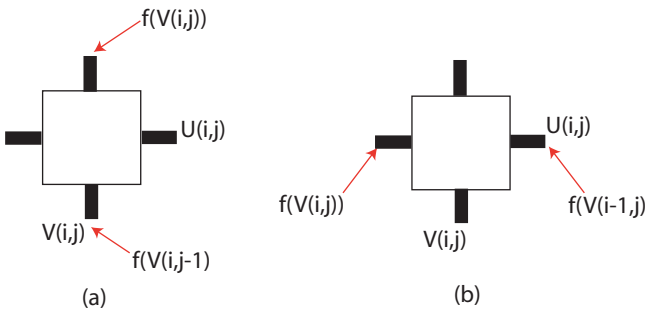


Figure 2.4: Illustration for the proof of Theorem 2

cannot be guaranteed to be wrong for incorrect value of $V(i,j)$. Hence, in at least one of the output pads an additional error checking portion $f(V(i,j))$ (that is input-sensitive to $V(i,j)$ and hence can reflect the error in $V(i,j)$) is required. It can be located on the top or left output pad.

- Assume that $f(V(i,j))$ is located on top pad, which implies $f(V(i,j-1))$ is located on the bottom pad (shown by arrows in Figure 2.4 (a)).

    1. If $V(i,j-1)$ does not exist within the input pads, then we need to consider the case when $f(V(i,j-1))$ has a mismatch. Since we require two further errors in the neighborhood of $T(i,j)$, as argued above it requires an additional error checking function $g(f(V(i,j-1)))$ (that is input-sensitive to $f(V(i,j-1))$) on at least one of the top or left output pad.

    2. If $V(i,j-1)$ exists in the input pads, then in case when $V(i,j-1)$ is mismatched, and two further errors in the neighborhood of $T(i,j)$ are required, it needs an additional error checking function $g'(V(i,j-1))$ (that is input-sensitive to $V(i,j-1)$) on at least one of the top or left output pad.

- Assume that $f(V(i,j))$ is located on left pad, which implies $f(V(i-1,j))$ is located on the right pad (shown by arrows in Figure 2.4 (b)).

    1. If $V(i-1,j)$ does not exist within the input pads, we need to consider the case

when $f(V(i-1,j))$ is mismatched. Since two further errors are required, as argued above it requires an additional error checking function $h(f(V(i-1,j)))$ (that is input-sensitive to $f(V(i-1,j))$) to be located on at least one of the top or left output pad.

2. If $V(i-1,j)$ exists in the input pads, then in case when $V(i-1,j)$ is mismatched, and two further errors are required, it requires an additional error checking function $h'(V(i-1,j))$ (that is input-sensitive to $V(i-1,j)$) to be present on at least one of the top or left output pads.

Hence, an additional error checking pad ($g(f(V(i,j-1)))$, $g'(V(i,j-1))$ or $h(f(V(i-1,j)))$or $h(V(i-1,j))$) is required on at least one of the output pads. Arguing in the same manner as above we conclude that this cycle will keep on repeating. Hence, it is not possible to construct a tile with a bounded number of parameters in the pads such that a mismatch results in two more mismatches in the neighborhood of the tile in a two-dimensional assembly. Combining it with Proposition 2.2.2 we conclude that redundancy based compact error resilient schemes can not reduce error from ϵ to ϵ^3. □

However, it will be proved that for a rather restricted class of Boolean functions OP_1 and OP_2, error can be reduced to ϵ^3 by using the construction of Figure 2.1 b), which is stated as Theorem 3.

Theorem 3. *For Boolean functions OP_1 and OP_2 which are pairwise input-sensitive, there exists a redundancy based compact error resilience scheme that can reduce the error to ϵ^3.*

Proof. For our proof, we will use the scheme shown in Figure 2.1 b). If OP_1 and OP_2 are restricted to be as described, and if the neighborhood tiles that are a-independent of $T(i,j)$ are assembled correctly, then a pad binding error in any of the input pads in $T(i,j)$ causes two additional mismatch errors in its neighborhood. As explained earlier, we need

to consider only the cases where the pad binding error occurs in either the bottom or the right pad of tile T(i,j). The following case analysis provides the required proof.

1. If the bottom pad of $T(i, j)$ has a mismatch:

   (a) If $V(i, j)$ in bottom pad of $T(i, j)$ has a mismatch, then the $V(i, j)$ in the right pad of $T(i, j)$ is incorrect. This causes a mismatch because according to our assumption, all the tiles a-independent of $T(i, j)$ are assembled correctly. Also:

      i. If $U(i, j)$ on right pad is correct, $V(i, j + 1)$ on top pad is incorrectly computed because of restrictions on OP_1 and OP_2. This will cause further mismatch at the right or bottom pad of the top neighbor $T(i, j + 1)$, as argued in the proof of Theorem 1.

      ii. If $U(i, j)$ on right pad has a pad-mismatch, then at least one of the $V(i, j + 1)$ on top pad or $U(i + 1, j)$ on left pad is incorrectly computed, because of the restrictions on OP_1 and OP_2. This will cause a further mismatch at right or bottom pad of the left neighbor ($T(i + 1, j)$) or top neighbor($T(i, j + 1)$) in the same way as argued earlier.

   (b) If $V(i, j)$ on bottom pad is correct and $V(i+1, j)$ on bottom pad has mismatch, then $V(i + 1, j)$ on the left pad is incorrect, which causes a further mismatch in the right or bottom pad of the left neighbor $T(i + 1, j)$. Also:

      i. If $U(i, j)$ on right pad is incorrect, then this causes a mismatch on the right pad of $T(i, j)$, because according to our assumption, all the tiles a-independent of $T(i, j)$ are assembled correctly.

      ii. If $U(i, j)$ on right pad is correct, then $U(i + 1, j)$ on left output pad is correct. But since $V(i + 1, j)$ has a mismatch, $V(i + 1, j + 1)$ on the top pad is incorrectly computed, because of the restriction on OP_1 and OP_2. This causes a further mismatch on the bottom or the right pad of the top

neighbor tile $T(i, j+1)$.

2. If there is no error in the bottom pad and there is mismatch in right pad:

   (a) If $U(i, j)$ on the right pad has a pad-mismatch, then at bottom $U(i, j)$ is incorrect, and causes a mismatch. However since $V(i, j)$ is correct on the bottom pad so $U(i+1, j)$ on the left pad is incorrectly computed because of the restriction on OP_1 and OP_2. This causes a further mismatch on right or bottom pad of left neighbor as explained earlier.

   (b) If $U(i, j)$ on right pad is correct and $U(i, j+1)$ has a mismatch, then $U(i, j+1)$ on top pad is incorrect, which causes a further mismatch in right or bottom pad of the top neighbor tile $T(i, j+1)$. Also since $V(i, j)$ is correct, $V(i, j+1)$ is also correct, and hence $U(i+1, j+1)$ on left pad is incorrectly computed because of restriction on OP_1 and OP_2. This causes a further mismatch in the right or bottom pad of the left neighboring tile $T(i+1, j)$.

Hence any mismatch on the right or bottom side of the tile $T(i, j)$ causes two further mismatches in the vicinity of tile $T(i, j)$. This results in error reduction from ϵ to ϵ^3 using Proposition 2.2.2. □

2.2.5 Error reduction to ϵ^4

Theorem 4. *For any Boolean functions OP_1 and OP_2, there exists no redundancy based compact error correction scheme that can reduce error from ϵ to ϵ^4 in two-dimensional self-assembly.*

Proof. For the reduction of error from ϵ to ϵ^4, a mismatch in any input pad should cause 3 more mismatches. It means that for any error in one of the input pads both the output pads should have errors. In case an output pad requires any additional error checking portion to

detect an error in an input, then by arguments similar to the proof of Theorem 2, it can be shown that such a tile cannot be constructed.

Hence, the only possibility is when, the left and top outputs $U(i+1,j)$ and $V(i,j+1)$ both change for any change in the input $U(i,j)$ or $V(i,j)$. This means that we have different values for each of $U(i+1,j)$ and $V(i,j+1)$ for 4 different values of input pair, which is not possible as $U(i+1,j)$ and $V(i,j+1)$ are Booleans. □

2.3 Error Correction in self-assemblies in three dimensions

Three dimensional self-assembly is being described as the most promising tool for heterogeneous integration of next generation microsystems. Its potential to build complex systems from microscale templates can not be overlooked[198, 113, 49, 187]. Besides the assembled three-dimensional structures can be extremely useful in computations[82]. It is possible to simulate a two-dimensional cellular automata, using three-dimensional self-assembly, which then paves way to perform a rich class of computations including matrix multiplication, integer multiplications, context-free language recognition etc.

2.3.1 Assembly in three dimensions

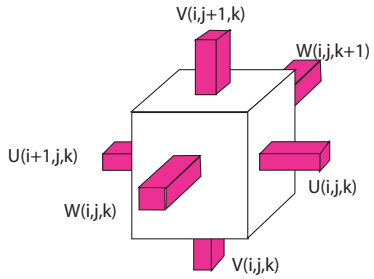


Figure 2.5: Three dimensional algorithmic self-assembly

The assembly problem in three-dimensions can be generalized from the two-dimensional

assembly as the assembly of a three-dimensional Boolean array of size $N \times M \times P$, where the elements are indexed from 0 to $N-1$ from right to left, 0 to $M-1$ from bottom to top, and 0 to $P-1$ from front to back. The bottommost horizontal plane, and rightmost and frontmost vertical planes provide the inputs to the assembly.

Let $V(i,j,k)$ be the value of element at i-th position from right, j-th position from bottom, and k-th position from front. Let $U(i+1,j,k)$ be the value communicated to the position $(i+1,j,k)$, $V(i,j+1,k)$ be communicated to the position $(i,j+1,k)$, and $W(i,j,k+1)$ be communicated to the position $(i,j,k+1)$. We define $U(i+1,j,k) = f_1(U(i,j,k),V(i,j,k),W(i,j,k))$, $V(i,j+1,k) = f_2(U(i,j,k),V(i,j,k),W(i,j,k))$, $W(i,j,k+1) = f_3(U(i,j,k),V(i,j,k),W(i,j,k))$ for Boolean functions f_1, f_2, and f_3.

Figure 2.5 shows a computational tile that can be used for construction of three-dimensional assembly. Right, bottom and front pads are the input pads, while the pads on left, top and back are output pads. As in two-dimensional assembly, a pad matches with the neighbor's contiguous pad if the values communicated by these pads are the same. $U(i,j,k)$, $V(i,j,k)$ and $W(i,j,k)$ are the right, bottom and front input pads, respectively, to the tile located at position (i,j,k). Then $U(i+1,j,k)$, $V(i,j+1,k)$ and $W(i,j,k+1)$ are the left, top and back output pads, respectively, of the tile $T(i,j,k)$. Also,$U(i+1,j,k) = f_1(U(i,j,k),V(i,j,k),W(i,j,k)), V(i,j{+}1,k) = f_2(U(i,j,k),V(i,j,k),W(i,j,k)), W(i,j,k{+}1) = f_3(U(i,j,k),V(i,j,k),W(i,j,k))$ where f_1, f_2 and f_3 are the ternary Boolean functions that take as input three Boolean values and give a Boolean output. It is assumed that initially a frame is assembled, with $M \times P$ tiles in rightmost plane, $N \times P$ tiles in bottommost plane and $N \times P$ tiles in frontmost plane. Next we examine the error resilience in three-dimensional self-assembly.

2.3.2 The Error Model

We extend the error model in two-dimensions to three-dimensional assembly in an obvious way. We follow the *independent error model* for three dimensional assembly. We also want to emphasize on the correct assembly of all the tiles in the assembly (and hence on the correctness of complete pattern), and not just on the correctness of final output only. We want to emphasize that the error analysis is done in the equilibrium state of the assembly. Consider a tile $T(i, j, k)$ in a $N \times M \times P$ tiling assembly where $0 < i < N - 1$, $0 < j < M - 1$, and $0 < k < P - 1$. We define the *immediate neighborhood* of a tile $T(i, j, k)$ as 26 tiles surrounding it, whose coordinates differ from (i, j, k) by at most 1. Formally speaking, $\{T(i', j', k') : |i' - i| \leq 1, |j' - j| \leq 1, |k' - k| \leq 1\} \setminus \{T(i, j, k)\}$. Tile $T(i', j', k')$ is said to be *a-dependent* on tile $T(i, j, k)$ if $i' \geq i$, $j' \geq j$, and $k' \geq k$ and *a-independent* otherwise. Next we examine the schemes to reduce the errors in self-assembly. As mentioned earlier *redundancy based compact error resilient scheme* refers to an error resilient scheme that does not scale up the assembly and in which the encodings on the pads of the tiles are used to create redundancy.

2.3.3 Error Reduction to ϵ^2

Theorem 5. *There exists a redundancy based compact error resilient tiling scheme in three dimensional assembly which can reduce the error from ϵ to ϵ^2 for any arbitrary Boolean functions f_1, f_2, and f_3, and it is shown in Figure 2.6.*

Proof. **Construction** Before we describe the construction, we would like to emphasize on the wholeness of pad. Each side of the tile has one pad in Figure 2.6, that encodes a 5-tuple as shown in the Figure. Disagreement between corresponding elements of two such 5-tuples in any two pads results in the total mismatch between those two pads. Consider the tile $T(i, j, k)$ with inputs $U(i, j, k)$, $V(i, j, k)$ and $W(i, j, k)$ on the right, bottom and

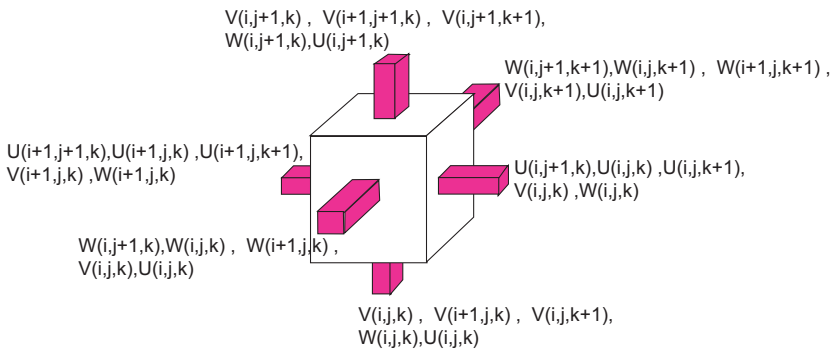


Figure 2.6: Construction for error reduction to ϵ^2

front pads respectively. Our goal is to guarantee one more error in the vicinity of this tile if there is one error in any of the input pads.

We add error checking portions to the right, bottom and front pads as shown in the Figure 2.6: $V(i,j,k)$ and $W(i,j,k)$ on right pad, $W(i,j,k)$ and $U(i,j,k)$ on bottom pad and $U(i,j,k)$ and $V(i,j,k)$ on front pad. Corresponding to these, we need to add $V(i+1,j,k)$ and $W(i+1,j,k)$ on left pad, $W(i,j+1,k)$ and $U(i,j+1,k)$ on top pad and $U(i,j,k+1)$ and $V(i,j,k+1)$ on back pad, as explained in the case of two-dimensional tile.

As described in two-dimensional assembly, every value in the output pads should be uniquely derivable from the values on the input pads. For $V(i+1,j,k)$ and $W(i+1,j,k)$ on the left pad we add $V(i+1,j,k)$ on the bottom pad, and $W(i+1,j,k)$ on the front pad. For $U(i,j+1,k)$ and $W(i,j+1,k)$ on the top pad, we add $U(i,j+1,k)$ to the right pad and $W(i,j+1,k)$ to the front pad. For $U(i,j,k+1)$ and $V(i,j,k+1)$ on the back pad, we add $U(i,j,k+1)$ to the right pad and $V(i,j,k+1)$ to the bottom pad. The construction is complete with addition of $U(i+1,j+1,k)$ and $U(i+1,j,k+1)$ to left pad, $V(i+1,j+1,k)$ and $V(i,j+1,k+1)$ to top pad, and $W(i+1,j,k+1)$ and $W(i,j+1,k+1)$ to back pad.

This completes our description of a tile in our compact error correction scheme. It

should be noted that the number of different tile types in this tile set will be 64 times as compared to number of tiles in a tile set without any error-correction. It can be attributed to the two values for each of the $U(i, j+1, k)$, $U(i, j, k+1)$, $V(i+1, j, k)$, $V(i, j, k+1)$, $W(i+1, j, k)$ and $W(i, j+1, k)$, for every value of the inputs $U(i, j, k)$, $V(i, j, k)$ and $W(i, j, k)$.

Error-Analysis: We show that if the neighborhood tiles a-independent of $T(i, j, k)$ are assembled correctly then a pad binding error in any of the input pads in $T(i, j, k)$ causes at least one additional mismatch error in its neighborhood in equilibrium. We need to consider only the cases where the pad binding error occurs in the bottom, the right or the front pad of tile $T(i, j, k)$. Otherwise, if the error occurs in top, left or back pad of $T(i, j, k)$ then we can consider the right pad of $T(i+1, j, k)$, bottom pad of $T(i, j+1, k)$ or front pad of $T(i, j, k+1)$, respectively, for the analysis. The following case analysis provides the required proof.

1. If the bottom pad of $T(i, j, k)$ has a mismatch:

   (a) If $V(i, j, k)$ on the bottom pad has a mismatch, then the values of $V(i, j, k)$ on the right and front pads are incorrect, which causes mismatches in right and front pads of $T(i, j, k)$.

   (b) If $V(i, j, k)$ on the bottom pad is correct and $V(i+1, j, k)$ on bottom pad of $T(i, j, k)$ has a mismatch, then $V(i+1, j, k)$ on left pad is incorrect. Now we will prove that it causes a further mismatch by exactly same technique as used by Reif et al[137]. We have assumed that all the tiles that are a-independent of tile $T(i, j, k)$ are correctly assembled so $T(i+1, j-1, k)$ is correctly assembled and has correct values of its top output pad. Hence $T(i, j, k)$'s left neighbor $T(i+1, j, k)$ is dependent upon the incorrect value communicated by the left pad of $T(i, j, k)$ and correct values communicated by top pad of $T(i+1, j-1, k)$. Now consider the pads of $T(i+1, j, k)$. The right pad includes $U(i+1, j+$

$1,k), U(i+1,j,k), U(i+1,j,k+1), V(i+1,j,k), W(i+1,j,k)$ and bottom pads include $V(i+1,j,k), V(i+2,j,k), V(i+1,j,k+1), W(i+1,j,k), U(i+1,j,k)$. Since the value $V(i+1,j,k)$ communicated by $T(i+1,j-1,k)$ is correct and the value $V(i+1,j,k)$ communicated by $T(i,j,k)$ is wrong, this implies there will be a mismatch at the right or bottom pad of Tile $T(i+1,j,k)$.

(c) If $V(i,j,k)$ and $V(i+1,j,k)$ on the bottom pad are correct and $V(i,j,k+1)$ on the bottom pad has a mismatch, then the value of $V(i,j,k+1)$ on the back pad is incorrect. Now we will show that it causes a further mismatch as argued above. Since we assume that all the tiles that are a-independent of tile $T(i,j,k)$ are correctly assembled so $T(i,j-1,k+1)$ is correctly assembled and has correct values of its top output pad. Hence $T(i,j,k)$'s back neighbor $T(i,j,k+1)$ is dependent upon the incorrect value communicated by the back pad of $T(i,j,k)$ and correct values communicated by top pad of $T(i,j-1,k+1)$. Now consider the pads of $T(i,j,k+1)$. The front pad includes $W(i,j+1,k+1), W(i,j,k+1), W(i+1,j,k+1), V(i,j,k+1), U(i,j,k+1)$ and bottom pads include $V(i,j,k+1), V(i+1,j,k+1), V(i,j,k+2), W(i,j,k+1), U(i,j,k+1)$. Since the value $V(i,j,k+1)$ communicated by $T(i,j-1,k+1)$ is correct and the value $V(i,j,k+1)$ communicated by $T(i,j,k)$ is wrong, this implies there will be a mismatch at the front or bottom pad of Tile $T(i,j,k+1)$.

2. If there is no error in bottom pad, but the right pad of $T(i,j,k)$ has mismatch:

   (a) If $U(i,j,k)$ on the right pad has a mismatch, then the values of $U(i,j,k)$ on bottom pad and front pad are incorrect, which causes two additional mismatches.

   (b) If $U(i,j,k)$ on right pad is correct but $U(i,j+1,k)$ on right pad is incorrect, then $U(i,j+1,k)$ on top output pad is incorrect. By the assumption of error-free assembly of a-independent tiles, we can argue as before that the value

$U(i, j+1, k)$ communicated by left pad of $T(i-1, j+1, k)$ is correct and the value $U(i, j+1, k)$ communicated by top pad of $T(i, j, k)$ is wrong, implying that there will be a mismatch at the right or bottom pad of Tile $T(i, j+1, k)$.

(c) If $U(i, j, k)$ and $U(i, j+1, k)$ on right pad are correct but $U(i, j, k+1)$ on right pad is incorrect, then $U(i, j, k+1)$ on back output pad is incorrect. By the assumption of error-free assembly of a-independent tiles, we can argue as before that the value $U(i, j, k+1)$ communicated by left pad of $T(i-1, j, k+1)$ is correct and the value $U(i, j, k+1)$ communicated by back pad of $T(i, j, k)$ is wrong, implying that there will be a mismatch at the right or front pad of Tile $T(i, j, k+1)$.

3. If there is no error in bottom or right pad of $T(i, j, k)$, but the front pad of $T(i, j, k)$ has mismatch:

   (a) If $W(i, j, k)$ on the front pad has a mismatch, then the values of $W(i, j, k)$ on bottom pad and right pad are incorrect, which causes two additional mismatches.

   (b) If $W(i, j, k)$ on front pad is correct but $W(i, j+1, k)$ on front pad is incorrect, then $W(i, j+1, k)$ on top output pad is incorrect. By the assumption of error-free assembly of a-independent tiles, we can argue as before that the value $W(i, j+1, k)$ communicated by back pad of $T(i, j+1, k-1)$ is correct and the value $W(i, j+1, k)$ communicated by top pad of $T(i, j, k)$ is wrong, implying that there will be a mismatch at the front or bottom pad of Tile $T(i, j+1, k)$.

   (c) If $W(i, j, k)$ and $W(i, j+1, k)$ on front pad are correct but $W(i+1, j, k)$ on front pad is incorrect, then $W(i+1, j, k)$ on left output pad is incorrect. By the assumption of error-free assembly of a-independent tiles, we can argue as before that the value $W(i+1, j, k)$ communicated by back pad of $T(i+1, j, k-$

$1)$ is correct and the value $W(i+1,j,k)$ communicated by left pad of $T(i,j,k)$ is wrong, implying that there will be a mismatch at the right or front pad of Tile $T(i+1,j,k)$.

Hence any mismatch on the right, bottom or front pad of tile $T(i,j,k)$ causes at least one more mismatch in the vicinity of the tile. Together with the Proposition 2.2.2 this implies that this scheme can reduce the pad mismatch errors from ϵ to ϵ^2. □

2.3.4 Error Reduction to ϵ^3

Theorem 6. *If Boolean functions f_1, f_2, and f_3 satisfy the following conditions:*

- *for fixed $V(i,j,k)$ and $W(i,j,k)$, $f_1(U,V,W)$ is input-sensitive to $U(i,j,k)$.*
- *for fixed $U(i,j,k)$ and $W(i,j,k)$, $f_2(U,V,W)$ is input-sensitive to $V(i,j,k)$.*
- *for fixed $U(i,j,k)$ and $V(i,j,k)$, $f_3(U,V,W)$ is input-sensitive to $W(i,j,k)$.*

Then there exists a compact error resilient scheme to reduce error from ϵ to ϵ^3 for three-dimensional self-assembly, and it is shown in Figure 2.6.

Proof. If f_1, f_2, and f_3 are as given in the theorem, and if the neighborhood tiles that are a-independent of $T(i,j,k)$ are assembled correctly, then a pad binding error in any of the input pads in $T(i,j,k)$ causes at least two additional mismatch errors in its neighborhood. As explained earlier, we need to consider only the cases where the pad binding error occurs in either the bottom, front or the right pad of tile $T(i,j)$. The following case analysis provides the required proof.

1. If the bottom pad of $T(i,j,k)$ has a mismatch:
   (a) If $V(i,j,k)$ on the bottom pad has a mismatch, then the values of $V(i,j,k)$ is incorrect in right and front pads, which causes two further mismatches in the right and front pads of $T(i,j,k)$.

(b) If $V(i,j,k)$ on the bottom pad is correct but $V(i+1,j,k)$ on the bottom pad of $T(i,j,k)$ has a mismatch, then the value of $V(i+1,j,k)$ on left pad is incorrect. This causes a mismatch in the right or bottom pad of the left neighboring tile $T(i+1,j,k)$, as argued earlier using the assumption of error-free assembly of a-independent tiles in the neighborhood of $T(i,j,k)$. Now there are two cases:

i. $U(i,j,k)$ in the right pad, $W(i,j,k)$ in the front pad, or $W(i+1,j,k)$ in the front pad of tile $T(i,j,k)$ has a mismatch. Thus, there is a further mismatches in the immediate neighborhood of tile $T(i,j,k)$.

ii. All three of the $U(i,j,k)$ in the right pad, and $W(i,j,k)$ and $W(i+1,j,k)$ in the front pad of tile $T(i,j,k)$ are correct. Since $V(i,j,k)$, $U(i,j,k)$, and $W(i,j,k)$ are correct in tile $T(i,j,k)$, $U(i+1,j,k)$ on the left pad is computed correctly. By the condition on f_1, f_2 and f_3, correct value of $U(i+1,j,k)$ and $W(i+1,j,k)$ and incorrect value of $V(i+1,j,k)$ results in incorrect computation of $V(i+1,j+1,k)$ on the top pad. By the assumption that tiles a-independent of $T(i,j,k)$ are assembled correctly, and as argued earlier, it can be proved that there will be another mismatch in the immediate neighborhood of $T(i,j,k)$.

(c) If $V(i,j,k)$ and $V(i+1,j,k)$ on the bottom pad are correct, but $V(i,j,k+1)$ on the bottom pad of $T(i,j,k)$ has a mismatch, then the value of $V(i,j,k+1)$ on the back pad is incorrect. This causes a mismatch in the front or bottom pad of the back neighboring tile $T(i,j,k+1)$. Now there are two cases:

i. $U(i,j,k)$ in the right pad, $U(i,j,k+1)$ in the right pad or $W(i,j,k)$ in front pad has a mismatch. Thus, there is a further mismatch in the immediate neighborhood of $T(i,j,k)$.

ii. All three of the $U(i,j,k)$ in the right pad, $U(i,j,k+1)$ in the right pad and $W(i,j,k)$ in front pad in tile $T(i,j,k)$ are correct. This results in

the correct computation of $W(i, j, k + 1)$. By the conditions on f_1, f_2 and f_3, correct value of $W(i, j, k + 1)$ and $U(i, j, k + 1)$ and incorrect $V(i, j, k + 1)$ results in the incorrect computation of $V(i, j + 1, k + 1)$ on the top pad. By the assumption that tiles a-independent of $T(i, j, k)$ are assembled correctly, and as argued earlier, it can be proved that there will be another mismatch in the immediate neighborhood of $T(i, j, k)$.

2. If there is no mismatch in the bottom pad of tile $T(i, j, k)$ but its right pad has a mismatch:

   (a) If $U(i, j, k)$ on the right pad has a mismatch, then the values of $U(i, j, k)$ is incorrect in bottom and front pads, which causes two further mismatches in the right and front pads of $T(i, j, k)$.

   (b) If $U(i, j, k)$ on the right pad is correct, but $U(i, j + 1, k)$ on the right pad of $T(i, j, k)$ has a mismatch, then the value of $U(i, j+1, k)$ on the top pad is incorrect. This causes a mismatch in the right or bottom pad of the top neighboring tile $T(i+1, j, k)$, due to the assumption of error-free assembly of a-independent tiles in the neighborhood of $T(i, j, k)$. Now there are two cases:

      i. $W(i, j, k)$ or $W(i, j+1, k)$ in the front pad of tile $T(i, j, k)$ has a mismatch. Thus, there is a further mismatches in the immediate neighborhood of tile $T(i, j, k)$.

      ii. Both $W(i, j, k)$ and $W(i, j+1, k)$ in the front pad of tile $T(i, j, k)$ are correct. Since $V(i, j, k)$, $U(i, j, k)$, and $W(i, j, k)$ are correct in tile $T(i, j, k)$, $V(i, j + 1, k)$ is computed correctly. By the condition on f_1, f_2 and f_3, correct value of $V(i, j + 1, k)$ and $W(i, j + 1, k)$ and incorrect value of $U(i, j + 1, k)$ results in incorrect computation of $U(i + 1, j + 1, k)$ on the left pad. By the assumption that tiles a-independent of $T(i, j, k)$ are as-

sembled correctly, and as argued earlier, it can be proved that there will be another mismatch in the immediate neighborhood of $T(i, j, k)$.

(c) If $U(i, j, k)$ and $U(i, j + 1, k)$ on the right pad are correct, but $U(i, j, k + 1)$ on the right pad of $T(i, j, k)$ has a mismatch, then the value of $U(i, j, k + 1)$ on the back pad is incorrect. This causes a mismatch in the front or bottom pad of the back neighboring tile $T(i, j, k + 1)$. Now there are two cases:

i. $W(i, j, k)$ in front pad has a mismatch. Thus, there is a further mismatch in the immediate neighborhood of $T(i, j, k)$.

ii. $W(i, j, k)$ in front pad in tile $T(i, j, k)$ is correct. This results in the correct computation of $W(i, j, k + 1)$. By the conditions on f_1, f_2 and f_3, correct value of $W(i, j, k+1)$ and $V(i, j, k+1)$ and incorrect $U(i, j, k+1)$ results in the incorrect computation of $V(i, j + 1, k + 1)$ on the top pad. By the assumption that tiles a-independent of $T(i, j, k)$ are assembled correctly, and as argued earlier, it can be proved that there will be another mismatch in the immediate neighborhood of $T(i, j, k)$.

3. If there is no error in the bottom pad or right pad of tile $T(i, j, k)$, but the front pad has a mismatch:

(a) If $W(i, j, k)$ on the front pad has a mismatch, then the values of $W(i, j, k)$ is incorrect in bottom and right pads, which causes two further mismatches in the bottom and right pads of $T(i, j, k)$.

(b) If $W(i, j, k)$ on the front pad is correct, but $W(i, j + 1, k)$ on the front pad of $T(i, j, k)$ has a mismatch, then the value of $W(i, j + 1, k)$ on the top pad is incorrect. This causes a mismatch in the front or bottom pad of the top neighboring tile $T(i+1, j, k)$, due to the assumption of error-free assembly of a-independent tiles in the neighborhood of $T(i, j, k)$. Since $V(i, j, k)$, $U(i, j, k)$,

and $W(i,j,k)$ are correct in tile $T(i,j,k)$, $V(i,j+1,k)$ is computed correctly. By the condition on f_1, f_2 and f_3, correct value of $V(i,j+1,k)$ and $U(i,j+1,k)$ and incorrect value of $W(i,j+1,k)$ results in incorrect computation of $U(i+1,j+1,k)$ on the left pad. By the assumption that tiles a-independent of $T(i,j,k)$ are assembled correctly, and as argued earlier, it can be proved that there will be another mismatch in the immediate neighborhood of $T(i,j,k)$.

(c) If $W(i,j,k)$ and $W(i,j+1,k)$ on the front pad are correct, but $W(i+1,j,k)$ on the front pad of $T(i,j,k)$ has a mismatch, then the value of $W(i+1,j,k)$ on the left pad is incorrect. This causes a mismatch in the right or bottom pad of the left neighboring tile $T(i+1,j,k)$. Correct values of $U(i,j,k)$, $V(i,j,k)$, and $W(i,j,k)$ results in the correct computation of $U(i+1,j,k)$. By the conditions on f_1, f_2 and f_3, correct value of $U(i+1,j,k)$ and $V(i+1,j,k)$ and incorrect $W(i+1,j,k)$ results in the incorrect computation of $V(i,j+1,k+1)$ on the top pad. By the assumption that tiles a-independent of $T(i,j,k)$ are assembled correctly, and as argued earlier, it can be proved that there will be another mismatch in the immediate neighborhood of $T(i,j,k)$.

Hence any mismatch on the bottom, right or front side of the tile $T(i,j,k)$ causes two further mismatches in the vicinity of tile $T(i,j,k)$ and this results in error reduction from ϵ to ϵ^3. □

2.3.5 Error Reduction to ϵ^4

Theorem 7. *For arbitrary Boolean functions f_1, f_2, and f_3, there exists no redundancy based compact error resilient scheme that can reduce error from ϵ to ϵ^4 in three-dimensional self-assembly.*

Proof. For errors to reduce from ϵ to ϵ^4, an error in any input pad, say $V(i,j,k)$ should cause three further mismatches in the immediate neighborhood. At least one of those

mismatches should be caused because of an error on one of the output pads. It should be noted that if the Boolean functions f_1, f_2 and f_3 are arbitrary Boolean functions then the outputs $U(i+1,j,k)$, $V(i,j+1,k)$ or $W(i,j,k+1)$

cannot be guaranteed to be wrong for incorrect value of $V(i,j,k)$. Hence, in at least one of the output pads an additional error checking portion $f(V(i,j,k))$ (that is input-sensitive to $V(i,j,k)$ and hence can reflect the error in $V(i,j,k)$) is required. It can be located on the top, left or back output pad.

- Assume that $f(V(i,j,k))$ is located on top side, which implies $f(V(i,j-1,k))$ is located on the bottom side.

    1. If $V(i,j-1,k)$ does not exist within the input pads, then we need to consider the case when $f(V(i,j-1,k))$ has a mismatch with bottom neighbor. Since we require this mismatch to cause three further errors in the neighborhood of $T(i,j,k)$, as argued above it requires an additional error checking function $g_1(f(V(i,j-1,k)))$ (that is input-sensitive to $f(V(i,j-1,k))$) to be located on at least one of the top, left, or back output pad.

    2. If $V(i,j-1,k)$ exists in the input pads, then in case when $V(i,j-1,k)$ is mismatched, and three further errors in the neighborhood of $T(i,j,k)$ are required, it needs an additional error checking function $g_1'(V(i,j-1,k))$ (that is input-sensitive to $V(i,j-1)$) to be located on at least one of the top, left or back output pad.

- Assume that $f(V(i,j,k))$ is located on left side, which implies $f(V(i-1,j,k))$ is located on the right side.

    1. If $V(i-1,j,k)$ does not exist within the input pads, we need to consider the case when $f(V(i-1,j,k))$ is mismatched. Since three further errors are required, as argued above it requires an additional error checking function $g_2(f(V(i-$

$1, j, k)))$ (that is input-sensitive to $f(V(i-1, j, k))$) to be located on at least one of the top, left or back output pad.

2. If $V(i-1, j, k)$ exists in the input pads, then in case when $V(i-1, j, k)$ is mismatched, and three further errors are required, it requires an additional error checking function $g_2'(V(i-1, j, k))$ (that is input-sensitive to $V(i-1, j, k)$) to be present on at least one of the top, left or back output pads.

- Assume that $f(V(i, j, k))$ is located on the back side, which implies $f(V(i, j, k-1))$ is located on the front side.

1. If $V(i, j, k-1)$ does not exist within the input pads, we need to consider the case when $f(V(i, j, k-1))$ is mismatched. Since three further errors are required, as argued above it requires an additional error checking function $g_3(f(V(i, j, k-1)))$ (that is input-sensitive to $f(V(i, j, k-1))$) to be located on at least one of the top, left or back output pad.

2. If $V(i, j, k-1)$ exists in the input pads, then in case when $V(i, j, k-1)$ is mismatched, and three further errors are required, it requires an additional error checking function $g_3'(V(i, j, k-1))$ (that is input-sensitive to $V(i, j, k-1)$) to be present on at least one of the top, left or back output pads.

Hence, an additional error checking pad ($g_1(f(V(i, j-1, k)))$, $g_1'(V(i, j-1, k))$, $g_2(f(V(i-1, j, k)))$, $g_2'(V(i-1, j, k))$, $g_3(f(V(i, j, k-1)))$, or $g_3'(V(i, j, k-1))$) is required on at least one of the output pads. Arguing in the same manner as above it can concluded that this cycle will keep on repeating. Hence, it is not possible to construct tile with a bounded number of parameters in the pads and we conclude that redundancy based compact error resilient schemes can not reduce error from ϵ to ϵ^4. □

2.4 Self-Healing Tile Set for Three Dimensional Assembly

Winfree [192] provided the basis for studying self-healing in the self-assembly in a rigorous manner. We need to consider the repairability of a self-assembled structure in the face of a damage. A tile set is called *self-healing*, if at any point during error-free growth, when n tiles are removed, subsequent error free growth will repair the damage rapidly [192]. Winfree's scheme of correctly repairing the damage (hole) is by ensuring that the holes are filled in the original forward direction of the algorithmic assembly and there is no backward growth in the holes.

Winfree proposed constructions of self-healing tile sets for two dimensional algorithmic self-assembly by replacing a single tile by a 3×3 (for simple assemblies like sierpinsky triangles) , 5×5 (for general assemblies) and 7×7 (for additional robustness to nucleation errors) block. We have extended his constructions to three dimensions. Each three-dimensional tile is replaced by a $3 \times 3 \times 3$ block of three-dimensional tiles to convert a tile set for simple assemblies into a self-healing tile set.

Figure 2.7 shows a $3 \times 3 \times 3$ block of tiles that replaces a computational tile. The internal glues inside the block are all unique to that block. The tile set given in Figure 2.7 guarantees that if a complete block needs to regrow, then it has to start from frontmost, bottommost and rightmost corner. The tile to be placed at this corner is uniquely and correctly determined, by the assembled neighboring blocks. The corner tiles that have at least one output side facing towards another block should be assembled in the end after the assembly of all other tiles in the block, so that they can be determined uniquely from the inputs and not ambiguosly from the outputs. We omit the Figures of blocks showing the frame tile and seed tile, but they can be derived easily following the same logic. Similarly, Winfree's other constructions for self-healing in two-dimensions can also be extended to three dimensions to improve the self-healing tile set.

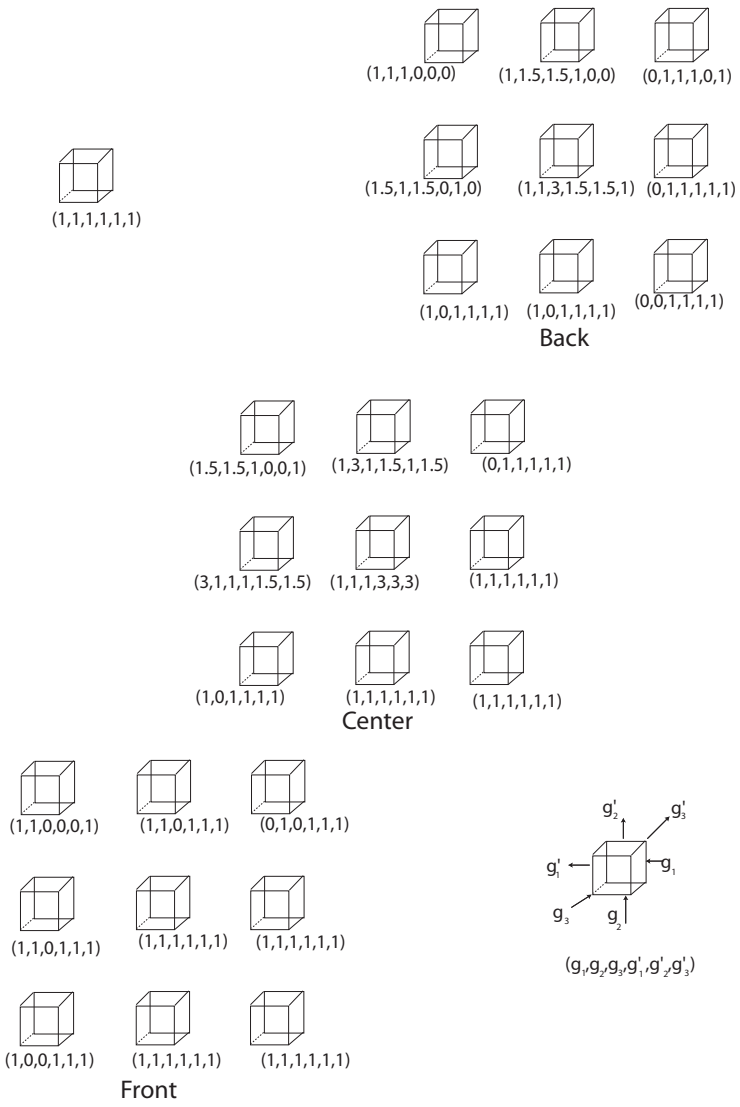


Figure 2.7: Self-healing tile set for three dimensional assembly showing only a computational tile. One tile is replaced by a $3 \times 3 \times 3$ block of tiles. The 6-tuple shown below every tile shows the glue-strengths for its sides. The order of the glue strengths in the tuple is as shown in the single tile in the top left portion of the Figure. The tuple $\langle g_1, g_2, g_3, g'_1, g'_2, g'_3 \rangle$ denotes the glue strength of g_1 on right, g'_1 on left, g_2 on bottom, g'_2 on top, g_3 on front and g'_3 on the back side of the tile. This construction corresponds to a computational tile in the assembly, which has the glue strength 1 on each of its 6 faces

2.5 Discussion

In this chapter, we presented a theoretical analysis of redundancy based compact error resilient tiling in two and three dimensions. We conjecture the following stronger results for

three-dimensional assemblies that are currently open questions to be proved or disproved. We state our conjectures as follows:

Conjecture: For arbitrary Boolean functions f_1, f_2, and f_3, there exists no redundancy based compact error correction scheme that will reduce error from ϵ to ϵ^3 in three-dimensional self-assembly. □

Conjecture: For any functions f_1, f_2, and f_3 that are outside the restricted class of the functions defined in Theorem 6 there exists no redundancy based compact error correction scheme that will reduce error from ϵ to ϵ^3 in three-dimensional self-assembly. □

Conjecture: For any Boolean functions f_1, f_2, and f_3, , there exists no redundancy based compact error resilient scheme that can reduce error from ϵ to ϵ^4 in three-dimensional self-assembly. □

The immediate future work will be to prove or disprove these conjectures. We have presented a three-dimensional extension to Winfree's self-healing tile set in two-dimensions. It remains an open question if it is possible to design a compact self-healing tile set for two and three-dimensional self-assembly.

Chapter 3

A Self-Assembly Model of Time-Dependent Glue Strength

Self-assembly is a process in which small objects autonomously associate with each other to form larger complexes. It is ubiquitous in biological constructions at the cellular and molecular scale and has also been identified by nanoscientists as a fundamental method for building nano-scale structures. It has aroused tremendous interest in mathematicians and computer scientists towards the theoretical study of the process of self-assembly. We propose a self-assembly model in which the glue strength between two juxtaposed tiles is a function of the time they have been in neighboring positions. We then present an implementation of our model using strand displacement reactions on DNA tiles.

We then study the tile complexity for assembling shapes in our model and show that a thin rectangle of size $k \times N$ can be assembled using $O(\frac{\log N}{\log \log N})$ types of tiles, for any constant $k > 0$. In addition to the minimization of the number of tile types for assemblies, there are various other interesting applications of our model including catalysis and self-replication. Catalysis is the phenomenon in which an external substance facilitates the reaction of other substances, without itself being used up in the process, and provides an alternative route of reaction where the activation energy is lower than the original chemical reaction and increase the reaction rate.

Under our model, we can demonstrate and study the process of catalysis in the tile assembly, and hence attempt to address the question posed by Adleman [4, 7]: Can we model the process of catalysis in self-assembly of tiles?

In addition to catalysis, our time-dependent glue model can also be used to model the process of self-replication of DNA tiles. Self-replication processes are one of the funda-

mental processes of nature, in which a system creates copies of itself, and from an engineering point of view, a material device that can self-replicate itself will be of great importance to achieve a low manufacturing cost.

3.1 Introduction

3.1.1 Motivation

Self-assembly is a ubiquitous process in which small objects self-organize into larger and complex structures. Examples in nature are numerous: atoms self-assemble into molecules, molecules into cells, cells into tissues, and so on. Recently, self-assembly has also been demonstrated as a powerful technique for constructing nano-scale objects. For example, a wide variety of DNA lattices made from self-assembled branched DNA molecules (DNA tiles) [41, 94, 104, 109, 125, 195, 201, 202] have been successfully constructed. Peptide self-assembly provides another nanoscale example [33]. Self-assembly is also used for mesoscale constructions using capillary forces [32, 146] or magnetic forces [1].

3.1.2 Prior Models for Tile Assembly

Mathematical studies of tiling dates back to 1960s, when Wang introduced his tiling model [183]. The initial focus of research in this area was towards the decidability/undecidability of the tiling problem [141]. A revival in the study of tiling was instigated in 1996 when Winfree proposed the simulation of computation [196] using self-assembly of DNA tiles.

In 2000, Rothemund and Winfree [148] proposed an *Abstract Tile Assembly (ATA) Model*, which is a mathematical model for theoretical studies of self-assembly. This model was later extended by Adleman *et al.* to include the time complexity of generating specified assemblies [5]. Later work includes combinatorial optimization, complexity problems, fault tolerance, and topology changes, in the abstract Tile Assembly Model as well as in

some of its variants [6, 8, 9, 42, 47, 48, 51, 63, 88, 89, 96, 137, 136, 147, 149, 154, 155, 167, 168, 192, 193].

Adleman introduced a reversible model [4], and studied the kinetics of the reversible linear self-assemblies of tiles. Winfree also proposed a kinetic assembly model to study the kinetics of the self-assembly [190]. Apart from these basic models, various generalized models of self-assembly are also studied [9, 86]: namely, multiple temperature model, flexible glue model, and q-tile model.

3.1.3 Needs for New Models for Tile Assembly

Though all these models contribute greatly towards a good understanding of the process of self-assembly, there are still a few things that could not be easily explained or modeled (for example, the process of catalysis and self-replication in tile assembly). Recall that catalysis is the phenomenon in which an external substance facilitates the reaction of other substances, without itself being used up in the process. A catalyst provides an alternative route of reaction where the activation energy is lower than the original chemical reaction and increase the reaction rate. Adleman [4] has posed an open question if we could model the process of catalysis in the self-assembly of tiles. Self-replication process is one of the fundamental process of nature, in which a system creates copies of itself. For example, DNA is self-replicated during cell division and is transmitted to offspring during reproduction. A material device that can self-replicate is ambition of many engineering disciplines. The biggest incentive is to achieve a low manufacturing cost because self-replication avoids the costs of labor, capital and distribution in conventional manufactured goods. In an evolving field like nanotechnology, manufacturing costs of molecular machines can become extremely large in the absence of self-replication. Recently, Schulman and Winfree show self-replication using the growth of DNA crystals [156], but their system requires shear forces to separate the replicated units. In this chapter we propose a new

model, in which catalysis and self-replication is possible without external intervention. In our new model, which is built on the basic framework of ATA Model, the glue strength between different glues is dependent on the time for which they have remained together.

The rest of the chapter is organized as follows. First we define the prior ATA Model as well as our new model formally in Section 3.2.2. We then put forth a method to physically implement such a system in Section 3.3. Then we present the processes of catalysis and self-replication in tile assembly in our model in Sections 3.4 and 3.5, respectively. In Section 3.6, we discuss the tile complexity of assembly of various shapes in our model, beginning with the assembly of thin rectangles in Section 3.6.1 and the extension to other shapes in Section 3.6.2. We conclude with the discussion of our results and future research directions in Section 3.7.

3.2 Tiling Assembly Models

3.2.1 The Abstract Tiling Assembly (ATA) Model

The *Abstract Tile Assembly (ATA) Model* was proposed by Rothemund and Winfree [148] in 2000. Intuitively speaking, a *tile* in the ATA model is a unit square where each side of the square has a *glue* from a set Σ associated with it. In this chapter we use the terms *pad* and side of the tile interchangeably. Formally, a tile is an ordered quadruple $(\sigma_n, \sigma_e, \sigma_s, \sigma_w) \in \Sigma^4$, where σ_n, σ_e, σ_s, and σ_w represent the *northern*, *eastern*, *southern*, and *western* side glues of the tile, respectively. Σ also contains a special symbol $null$, which is a zero-strength glue. T denotes the set of all tiles in the system. A tile cannot be rotated. So, $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \neq (\sigma_2, \sigma_3, \sigma_4, \sigma_1)$. Also defined are various projection functions $n : T \to \Sigma$, $e : T \to \Sigma$, $s : T \to \Sigma$, and $w : T \to \Sigma$, where $n(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \sigma_1$, $e(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \sigma_2$, $s(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \sigma_3$, and $w(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = \sigma_4$.

A glue strength function $g : \Sigma \times \Sigma \to \mathbb{R}$ determines the glue strength between two abutting tiles. $g(\sigma, \sigma') = g(\sigma', \sigma)$ is the strength between two tiles that abut on sides with

glues σ and σ'. If $\sigma \neq \sigma'$, $g(\sigma, \sigma') = 0$; otherwise it is a positive value. It is also assumed that $g(\sigma, null) = 0$, $\forall \sigma \in \Sigma$. In the tile set T, there is a specified unique *seed* tile s. There is a system parameter to control the assembly known as *temperature* and denoted as τ. All the ingredients described above constitute a *tile system*, a quadruple $\langle T, s, g, \tau \rangle$. A *configuration* is a snapshot of the assembly. More formally, it is the mapping from $\mathbb{Z}^2$ to $T \bigcup \{EMPTY\}$ where $EMPTY$ is a special tile $(null, null, null, null)$, indicating a tile is not present. For a configuration C, a tile $A = (\sigma_n, \sigma_e, \sigma_s, \sigma_w)$ is attachable at position (i, j) iff $C(i, j) = EMPTY$ and $g(\sigma_e, w(C(i, j+1))) + g(\sigma_n, s(C(i+1, j))) + g(\sigma_w, e(C(i, j-1))) + g(\sigma_s, n(C(i-1, j))) \geq \tau$, where indices i and j increase towards north and east directions, respectively.

Assembly takes place sequentially starting from a seed tile s at a known position. One key aspect of this ATA Model is that the glues are constant over time. For a given tile system, any assembly that can be obtained by starting from the *seed* and adding tiles one by one, is said to be *produced*. An assembly is called to be *terminally produced* if no further tiles can be added to it. The *tile complexity* of a shape S is the size of the smallest tile set required to uniquely and terminally assemble S under a given assembly model. One of the well-known results is that the tile complexity of self-assembly of a square of size $N \times N$ in ATA model is $\Theta(\frac{\log N}{\log \log N})$ [5, 148].

3.2.2 Our Time-Dependent Glue (TDG) Model

We propose a Time-dependent Glue Model, which is built on the framework described above. In this model, the glue-strength between two tiles is dependent upon the time for which the two tiles have remained together.

Let τ be the temperature of the system. Tiles are defined as in the ATA Model. However, in our model, glue strength function, g, is extended to contain a third argument that specifies the time for which the two sides of tiles are in contact. Formally speaking, g is

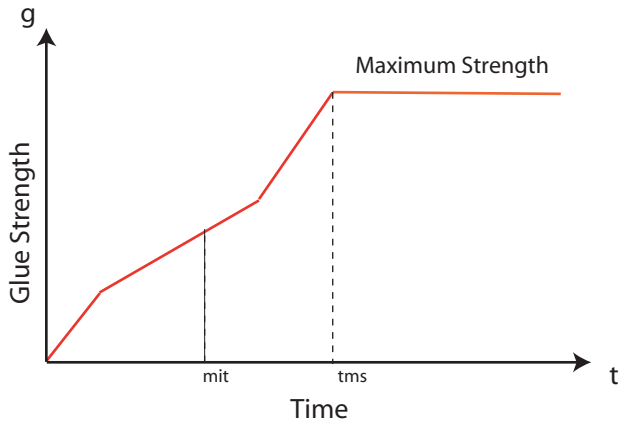


Figure 3.1: A graph that illustrates the concept of time-dependent glue strength, minimum interaction time, and time for maximum strength

defined as $g : \Sigma \times \Sigma \times \mathbb{R} \rightarrow \mathbb{R}$.

In $g(\sigma, \sigma', t)$ the argument t is the time for which two sides of the tiles with glue-labels σ and σ' have been juxtaposed. For every pair (σ, σ'), the value $g(\sigma, \sigma', t)$ increases with t up to a maximum limit and then takes a constant value determined by σ and σ'. We define the time when g reaches this maximum as *time for maximum strength* and denote it as $\gamma : \Sigma \times \Sigma \rightarrow \mathbb{R}$. Note $g(\sigma, \sigma', t) = g(\sigma, \sigma', \gamma(\sigma, \sigma'))$ for $t \geq \gamma(\sigma, \sigma')$.

The *minimum interaction time* is a function $\mu : \Sigma \times \Sigma \rightarrow \mathbb{R}$. For every pair (σ, σ'), a function $\mu(\sigma, \sigma')$ is defined as the minimum time for which the two tiles with abutting glue symbols σ and σ' stay together. If $g(\sigma, \sigma', \mu(\sigma, \sigma')) \geq \tau$, the two tiles will stay together; otherwise they will separate if there is no other force holding them in their abutting positions. An example of glue-strength function is shown in Figure 3.1. Intuitively speaking, μ serves as the minimum time required by the pads to decide whether they want to separate or remain joined. We further define $\mu(\sigma, null) = 0$, $\gamma(\sigma, null) = 0$, and $g(\sigma, null, t) = 0$.

Next we give the justification and estimation of μ for a pair (σ, σ') of glues. Let $g(\sigma, \sigma', t)$ be the glue strength function. For more realistic estimation of μ, consider a physical system in which, in addition to association, dissociation reactions also occur. Let

$p(b)$ be the probability of dissociation when the bond strength is b, where $p(b)$ can be determined using Winfree's kinetic model [190]. Assume that $f(t)$ be the probability that no dissociation takes place in the time interval $[0, t]$, and assume the time-interval δt is so small that bond strength $g(\sigma, \sigma', t)$ does not change in the time-interval t and $t + \delta t$. Then,

$$f(t + \delta t) = f(t) \cdot (1 - p(g(\sigma, \sigma', t)))^{\delta t}$$

$$\frac{f(t + \delta t)}{f(t)} = (1 - p(g(\sigma, \sigma', t)))^{\delta t}$$

$$\frac{f(t + \delta t)}{f(t)} = \exp(-\delta t \cdot p(g(\sigma, \sigma', t)))$$

The probability that the dissociation takes place between time t and $t + \delta t$ is given by $f(t) \cdot (1 - \exp(-\delta t \cdot p(g(\sigma, \sigma', t))))$. Since μ is defined as the time for which two glues are expected to remain together once they come in contact, its expected value is:

$$E[\mu] = \lim_{\delta t \to 0} \sum_{t=0}^{\infty} t \cdot f(t) \cdot (1 - \exp(-\delta t \cdot p(g(\sigma, \sigma', t))))$$

Hence, based on the knowledge of glue strength function it is possible to determine the expected minimum interaction time for a pair (σ, σ'). For simplicity, we will use the expected value of μ as the actual value of μ for a pair of glues (σ, σ').

Next we illustrate the time-dependent model with an example of the addition of a single tile to an aggregate. In a configuration C, when a position (i, j) becomes available for the addition of a tile A, it will stay at (i, j) for a time interval t_0, where t_0 = max $\{\mu(e(A), w(C(i, j+1)))$, $\mu(n(A), s(C(i+1, j)))$,$\mu(w(A), e(C(i, j-1)))$,$\mu(s(A), n(C(i-1, j)))\}$. Recall that our model requires that if two tiles ever come in contact, they will stay together till the minimum interaction time of the corresponding glues.

After this time interval t_0, if $g(e(A), w(C(i, j+1)), t_0) + g(n(A), s(C(i+1, j)), t_0) + g(w(A), e(C(i, j-1)), t_0) + g(s(A), n(C(i-1, j)), t_0) < \tau$, tile A will detach; otherwise,

A will continue to stay at position (i, j).

We describe in the next section a method to implement our model of time-dependent glue strength with DNA tiles.

3.3 Implementation of Time-Dependent Glue Model

In this Section, we propose an implementation of Time-Dependent Glue Model using DNA. Structurally, DNA is a long polymer of simple units called nucleotides, which are held together by a backbone made of sugars and phosphate groups. This backbone carries four types of bases (A, C, T and G). These bases form complementary pairs (A is complementary to T and C is complementary to G) in a sense that each base can form hydrogen bonds with the complementary base, also known as Watson-Crick base-pairing. The hydrogen bonding between complementary base pairs from two DNA strands results in their intertwining in the shape of a double helix, known as double stranded DNA (dsDNA). Individual separate strands are known as single stranded DNA (ssDNA). The direction of a DNA strand is defined in terms of its asymmetric ends, referred to as 5' and 3' ends. The 5' end terminates at the phosphate group attached to the fifth carbon atom in the sugar ring, while the 3' end terminates at hydroxyl group attached to the third carbon atom in the sugar-ring. In a double helix, direction of the nucleotides in one strand is opposite to their direction in the other strand.

The process of combining complementary, single stranded nucleic acids into a double stranded DNA molecule is called *DNA hybridization*. If the hydrogen bonds between the nucleotides in two hybridizing DNA strands build up sequentially, the total binding force between the two strands will increase with time up to the complete hybridization, which will provide a simple way of obtaining time-dependent glue strength between DNA tiles. However, even if we assume that the hybridization of two complementary DNA strands is instantaneous, we can design a multi-step binding mechanism to implement the idea of

time-dependent glue strength, which exploits the phenomenon of strand displacement.

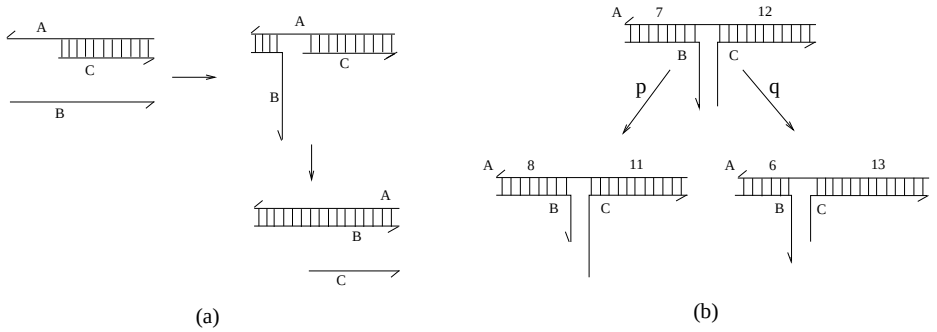


Figure 3.2: Figure (a) illustrates the process of strand displacement. Figure (b) shows a single step of strand-displacement as single step of random walk. In (b), the numbers represent the number of DNA base pairs

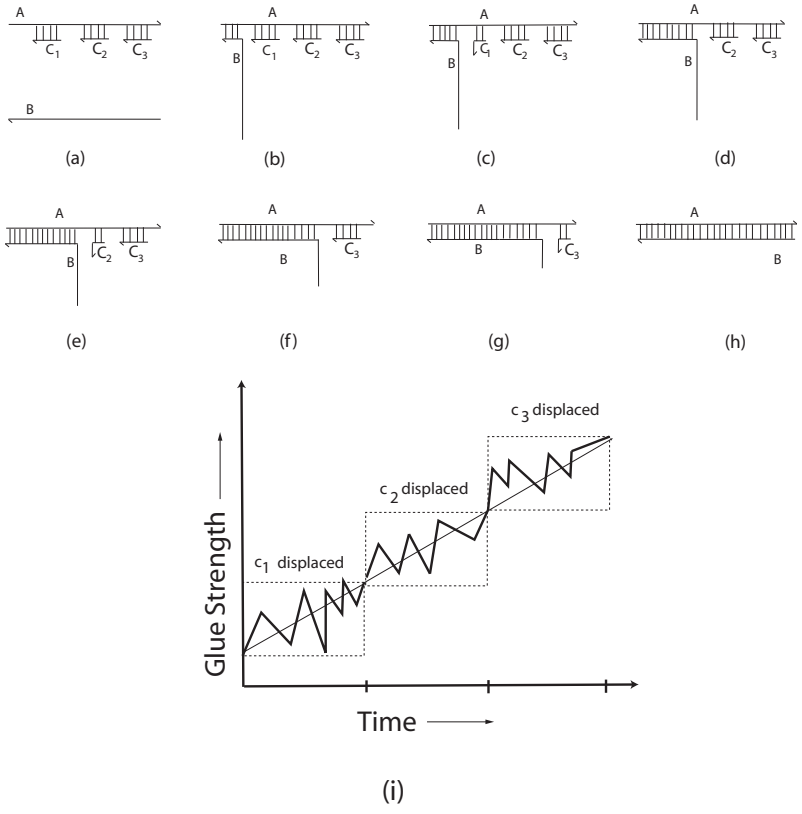


Figure 3.3: Figures (a) to (h) illustrate a mechanism by which strand displacement reaction is used to implement time-dependent glue between two pads. They show step by step removal of C_i's by B from A. In Figure 3.3 (i) an imaginary graph illustrates the variation of glue-strength between A and B w.r.t. time

Figure 3.2 (a) illustrates the process of strand displacement in which strand B displaces strand C from strand A. Figure 3.2 (b) illustrates one step during this process. At any time either the hybridization of B with A (and hence dehybridization of C from A) or hybridization of C with A (and hence dehybridization of B from A) can proceed with certain probability. Hence, we can model the strand displacement process as a random walk, with forward direction corresponding to hybridization between B and A, and backward direction corresponding to hybridization between C and A. A *one-dimensional unbiased random walk* is a process in which any step in forward or backward direction is taken with probability 0.5 independent of previous steps. The average straight-line distance between start and finish points of a one-dimensional random walk after n steps is on the order of $\sqrt{n}$, and hence expected number of steps to cover a distance n is $O(n^2)$ [57, 75, 139]. In order to model the strand displacement, we can assume that the step length in this random walk is 1 base-pair long. Hence, if the length of C is n bases, the expected number of steps required for B to replace C is $O(n^2)$.

Next we describe the design of the pads of DNA tiles with time dependent glue using the above mechanism of strand displacement. To make the glue between pad A and pad B time-dependent, we need a construction similar to the one in Figure 3.3 (a). The strand representing pad A has various smaller strands (C_i's, called *protector strands*) hybridized to it as shown in Figure 3.3 (a). The strand B will displace these protector strands C_i sequentially.

Let the variable γ here will be the time required for B to displace all the C_i's. In the case when there are k different small strands C_i of length n_i attached to A, γ is $\sum_{i=1}^{k} n_i^2$.

Figure 3.3 gives the step by step illustration of the above process. The variation of glue strength between A and B is shown in Figure 3.3 (i). By controlling the length of various C_i's (*i.e.* $n_1, n_2, \ldots, n_k$), we can control the glue-strength function g for a pair of tile-pads (or glues). Thus, we have shown a method to render the DNA tiles the characteristic of

time-dependent glue strength.

An interesting property is that the individual strand displacement of B against C_i is modeled as an unbiased one dimensional random walk, but the complete process described above can be viewed as *roughly* monotonic. As shown in Figure 3.3 (i), the strength of the hybridization between strand A and strand B increases in a roughly monotonic fashion with the removal of every C_i. However during the individual competition between B and C_i, the increase is not monotonic.

3.4 Catalysis

Catalysis is the phenomenon in which an external substance facilitates the reaction of other substances, without itself being used up in the process. A catalyst provides an alternative route of reaction where the activation energy is lower than the original chemical reaction and increase the reaction rate. Catalysts participate in reactions but are neither reactants nor products of the reaction they catalyze. The following question was posed by Adleman [4]: Can we model the process of catalysis in self-assembly of tiles? In this section, we present a model for catalysis in self-assembly of tiles using our time-dependent glue model. Now consider a supertile $\mathcal{X}$ (composed of two attached tiles C and D) and two single tiles A and B as shown in Figure 3.4 (a). We describe below how $\mathcal{X}$ can serve as a catalyst for the assembly of A and B. Assume $t_0 = \mu(e(A), w(B))$ such that $g(e(A), w(B), t_0)$ is less than the temperature τ. Let $\mu(s(A), n(C))$ = $\mu(s(B), n(D))$ = $t_1 > t_0$. Also assume $g(s(A), n(C), t_1) + g(s(B), n(D), t_1) < \tau$ and $g(e(A), w(B), t_1) \geq \tau$.

The graph in Figure 3.4 (b) illustrates an example set of required conditions for the glue strength functions in the system. $A \cdot B$ represents a tile A bounded to a tile B. To show that $\mathcal{X}$ acts as a catalyst, we first show that without $\mathcal{X}$ stable $A \cdot B$ can not form. Next we show that $A \cdot B$ will form when $\mathcal{X}$ is present and $\mathcal{X}$ will be recovered unchanged after the formation of $A \cdot B$.

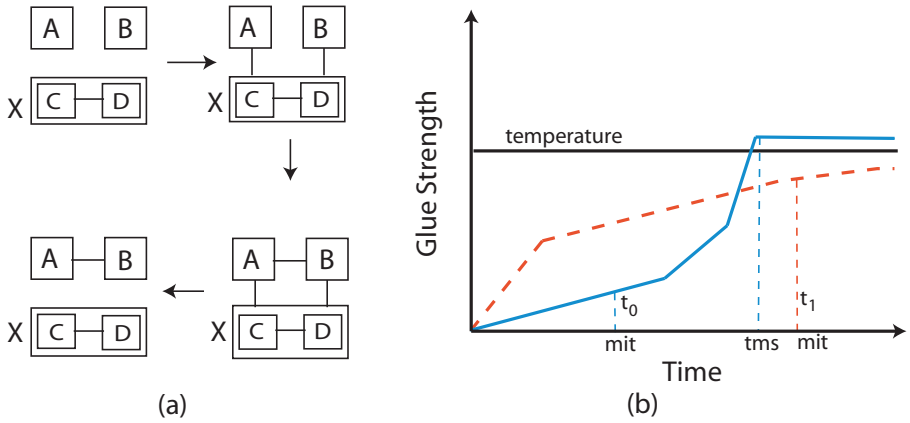


Figure 3.4: Figure (a) shows catalyst $\mathcal{X}$ with the tiles C and D catalyzes the formation of $A \cdot B$. (b) shows the conditions required for catalysis in terms of the glue strength function. Solid line shows the plot of $g(e(A), w(B), t)$ and dashed line shows the plot of $g(s(A), n(C), t) + g(s(B), n(D), t)$

Without $\mathcal{X}$ in the system, A and B can only be held in neighboring positions for time $t_0 = \mu(e(A), w(B))$, since $g(e(A), w(B), t_0) < \tau$. Hence, at t_0, A and B will fall apart.

However, in the presence of $\mathcal{X}$, the situation changes. Supertile $\mathcal{X}$ has two neighboring tiles C and D. Tiles A and B attach themselves to C and D as shown in Figure 3.4 (a). Since we let $\mu(s(A), n(C)) = \mu(s(B), n(D)) = t_1 > t_0$, tiles A and B are held in the same position for time t_1. By our construction, as shown in Figure 3.4 (b), the following two events will occur at time t_1:

- At t_1, the glue strength between A and B is $g(e(A), w(B), t_1) \geq \tau$ and hence A and B will be glued together. That is, in the presence of $\mathcal{X}$, A and B remain together for a longer time, producing stably glued $A \cdot B$.

- At t_1, the total glue strength between $A \cdot B$ and $\mathcal{X}$ is $g(s(A), n(C), t_1) + g(s(B), n(D), t_1) < \tau$, and the glued $A \cdot B$ will fall off $\mathcal{X}$. $\mathcal{X}$ is recovered unchanged from the reaction and the catalysis is complete. Now $\mathcal{X}$ is ready to catalyze other copies of A and B.

Note that if only A (resp. B) comes in to attach with C (resp. D), it will fall off at the end of time $\mu(s(A), n(C))$ (resp. $\mu(s(B), n(D))$). If assembled $A \cdot B$ comes in, it will also

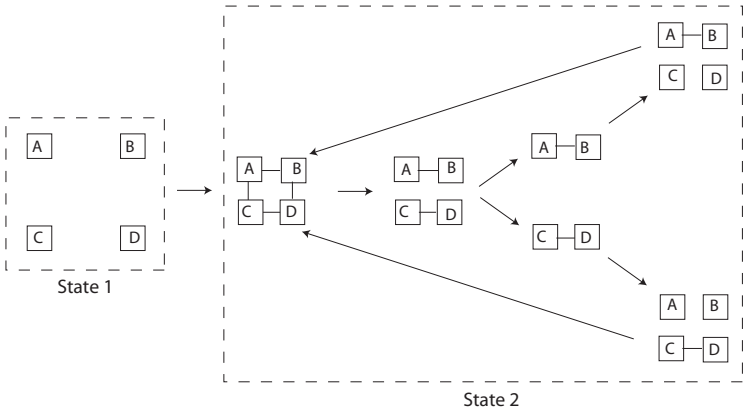


Figure 3.5: A schematic of self-replication

fall off, at time t_1. These two reactions are futile reactions, and do not block the desired catalysis reaction. However, as the concentration of $A \cdot B$ increases and the concentration of unattached A and B decreases, the catalysis efficiency of $\mathcal{X}$ will decrease due to the increased probability of the occurrence of futile reaction between $A \cdot B$ and $C \cdot D$.

3.5 Self-replication

Self-replication process is one of the fundamental process of nature, in which a system creates copies of itself. For example, DNA is self-replicated during cell division and is transmitted to offspring during reproduction. A material device that can self-replicate is ambition of many engineering disciplines. The biggest incentive is to achieve a low manufacturing cost because self-replication avoids the costs of labor, capital and distribution in conventional manufactured goods. In an evolving field like nanotechnology, manufacturing costs of molecular machines can become extremely large in the absence of self-replication. We discuss below an approach to model the process of self-replication in DNA tiles assembly using our time-dependent glue model.

Our approach is built on the above described process of catalysis: a product $A \cdot B$ catalyzes the formation of $C \cdot D$, which in turn catalyzes the formation of $A \cdot B$. And

hence an exponential growth of self-replicated $A \cdot B$ and $C \cdot D$ takes place.

More precisely, let $t_0 < t_1$, and consider tiles A, B, C, and D, such that :

$$\begin{aligned}
\mu(e(A), w(B)) = \mu(e(C), w(D)) &= t_0, \\
\mu(s(A), n(C)) = \mu(s(B), n(D)) &= t_1, \\
g(e(A), w(B), t_0) = g(e(C), w(D), t_0) &< \tau, \\
g(e(A), w(B), t_1) = g(e(C), w(D), t_1) &> \tau, \\
g(s(A), n(C), t_1) + g(s(B), n(D), t_1) &< \tau.
\end{aligned}$$

A system containing these four types of tiles has two states:

State 1. If there is no template $A \cdot B$ or $C \cdot D$ in the system, no assembled supertile exists since no two tiles can be held together long enough to form strong enough glue between them such that they become stably glued. Since $\mu(e(A), w(B)) = \mu(e(C), w(D)) = t_0$ and $g(e(A), w(B), t_0) = g(e(C), w(D), t_0) < \tau$, neither stable $A \cdot B$ nor stable $C \cdot D$ can form. Similarly, $\mu(s(A), n(C)) = \mu(s(B), n(D)) = t_1$, $g(s(A), n(C), t_1) < \tau$, and $g(s(B), n(D), t_1) < \tau$ implies that neither stable $A \cdot C$ nor stable $B \cdot D$ can form.

State 2. In contrast, if there is an initial copy of stable $A \cdot B$ in the system, self-replication occurs as follows. $A \cdot B$ serves as catalyst for the formation of $C \cdot D$, and $C \cdot D$ and $A \cdot B$ separate from each other at the end of the catalysis period, as described in Section 3.4; in turn, $C \cdot D$ serves as catalyst for the formation of $A \cdot B$. Thus we have a classical self-replication system: one makes a copy of itself via its complement. The number of the initial template ($A \cdot B$) and its complement ($C \cdot D$) grows exponentially in such a system as long as there are sufficient numbers of free A, B, C and D tiles to be made into pairs.

Hence, if the system is in state 1, it needs a triggering activity (formation of a stable $A \cdot B$ or $C \cdot D$) to go into state 2. Once the system is in state 2, it starts the self-replication process. Figure 3.5 illustrates the process of self-replication in the assembly of tiles.

If the system is in state 1, then the triggering activity (formation of a stable $A \cdot B$ or $C \cdot D$) can take place only if A, B, C, D co-position themselves so that the east side of A faces the west side of B and the south side of A faces the north side of C, and at the same time the south side of B faces the north side of D. In such a situation, A and C will remain abutted till time t_1, B and D will remain abutted till time t_1, and A and B (and C and D) might also remain together for time t_1, producing stable $A \cdot B$ and stable $C \cdot D$. And this will bring the system to state 2. Such copositioning of 4 tiles is a very low probability event. However, among other conditions, appropriate copositioning of unstable $A \cdot B$ and unstable $C \cdot D$, or unstable $A \cdot C$ and unstable $B \cdot D$ can also perturb a system in state 1 and triggers tremendous changes by bringing the system to state 2 where self-replication occurs.

3.6 Tile Complexity Results

3.6.1 Tile complexity results for thin rectangles

In the ATA Model, the tile complexity of assembling an $N \times N$ square is $\Theta(\frac{\log N}{\log \log N})$ [5, 148]. It is also known that the upper bound on the tile complexity of assembling a $k \times N$ rectangle in the ATA Model is $O(k + N^{1/k})$ and that the lower bound on tile complexity of assembling a $k \times N$ rectangle is $\Omega(\frac{N^{1/k}}{k})$ [9]. For small values of k this lower-bound is asymptotically larger than $O(\frac{\log N}{\log \log N})$. Here we claim that, in our model, as in the multi-temperature model defined in [9], a $k \times N$ rectangle can be self-assembled using $O(\frac{\log N}{\log \log N})$ types of tiles, even for small values of k. The proof technique follows the same spirit as in [9].

Theorem 8. *In time-dependent glue model, the tile complexity of self-assembling a $k \times N$ rectangle for an arbitrary integer $k \geq 2$ is $O(\frac{\log N}{\log \log N})$.*

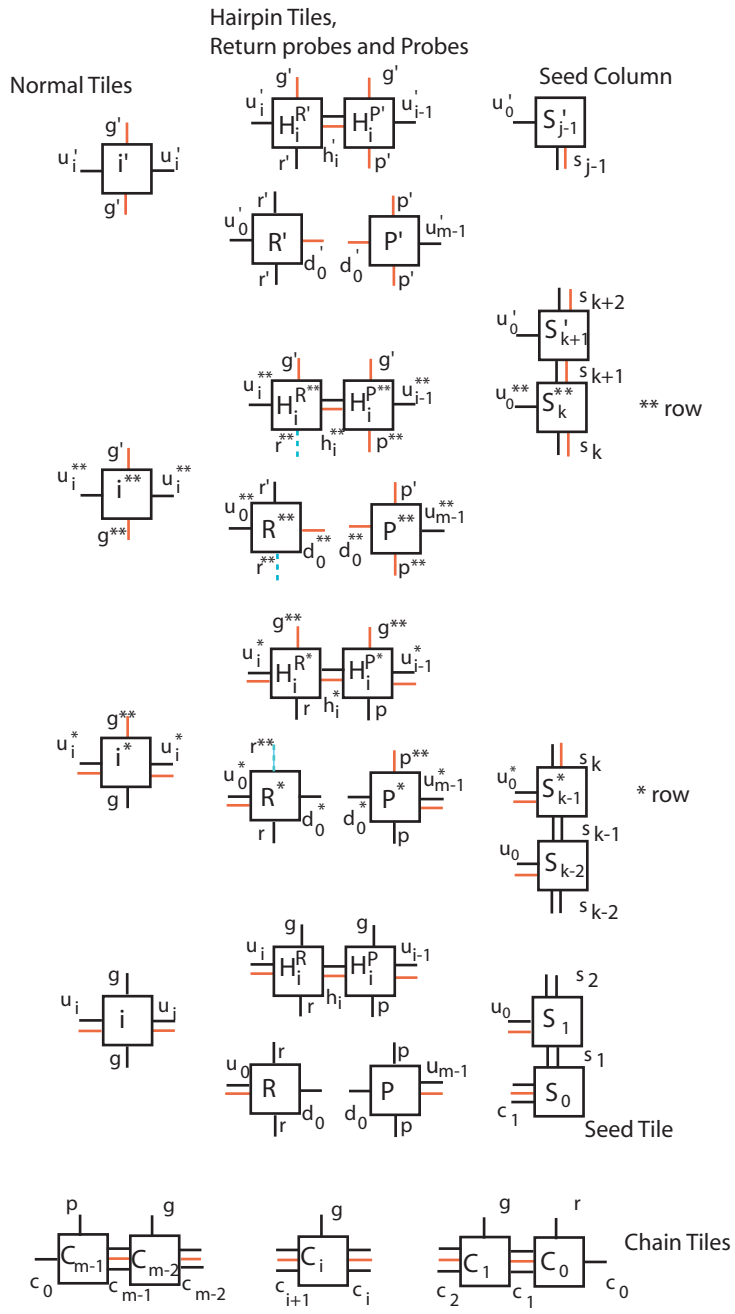


Figure 3.6: Tile set to construct a $k \times N$ rectangle using only $O(N^{1/j} + j)$ tiles. The glue strength functions of gray, dashed, and black glues are defined in the proof

Proof. The tile complexity of self-assembling a $k \times N$ rectangle is $O(N^{\frac{1}{k}} + k)$ for the ATA Model [9]. In time dependent glue model, we can use the similar idea as in [9] to reduce the tile complexity of assembling thin rectangles. For given k and N, build a $j \times N$ rectangle with $j > k$ such that the glues among the first k rows become strong after their μ (*minimum interaction time*), while the glues among the last $j - k$ rows do not become as strong. First k rows are called *stable rows* and last $j - k$ rows are called *volatile rows*. As such, these $j - k$ volatile rows, will disassemble from the assembly after certain time leaving the target $k \times N$ rectangle consisting of only the stable rows.

The tile set required to accomplish this construction is shown in Figure 3.6, which is similar to the one used in [9]. For more detailed illustration of this tile set, refer to [9]. First, a j-digit m-base counter is assembled as follows. Starting from the west edge of the seed tile, a chain of length m is formed in the first row using m chain tiles. At the same time tiles in the seed column also start assembling. It should be noted that first k tiles in the seed column have sufficient glue-strength and they are stable. Now starting from their west edges, the 0 normal tiles start filling the $m - 1$ columns in the upper rows. Then the hairpin tiles H_1^P and H_1^R assemble in the second row, which causes the assembly of further m chain tiles in the first row, and the assembly of 1 normal tiles in the second row (and 0 normal tiles in the upper rows) in the next section of m columns. Generally speaking, whenever a C_{m-1} chain tile is assembled in the first row, probe tiles in the upper rows are assembled until reaching a row that does not contain an $m - 1$ normal tile. In such a row, the appropriate hairpin tiles are assembled and this further propagates the assembly of return probe tiles downwards until the first row is reached, where a C_0 chain tile gets assembled. This again starts an assembly of a chain of length m. The whole process is repeated until a $j \times m^j$ rectangle is assembled.

Next we describe our modifications which are required for the $j-k$ upper volatile rows to get disassembled after the complete assembly of the $j \times m^j$ rectangle. First of all we

need to have a special $(k+1)$-th row ($**$ row), which will assemble to the north of the k-th row ($*$ row), as shown in Figure 3.6.

The operating temperature $\tau = 2$. Assume that for all glue-types, $\mu = t_0$ and $\gamma = t_1$. There are three kinds of glues shown in Figure 3.6: black, gray, and dashed. Assume that the glue-strength function for a single black glue is $g_{\text{black}}(t)$, a single gray glue is $g_{\text{gray}}(t)$, and a single dashed glue is $g_{\text{dashed}}(t)$. They are defined as

$$g_{\text{black}}(t) = \begin{cases} \frac{4t}{5t_0} & t < t_0 \\ \frac{4}{5} + \frac{t-t_0}{5(t_1-t_0)} & t_0 \leq t < t_1 \\ 1 & t \geq t_1 \end{cases}$$

$$g_{\text{gray}}(t) = \begin{cases} \frac{2t}{5t_0} & t < t_0 \\ \frac{2}{5} + \frac{t-t_0}{10(t_1-t_0)} & t_0 \leq t < t_1 \\ \frac{1}{2} & t \geq t_1 \end{cases}$$

$$g_{\text{dashed}}(t) = \begin{cases} \frac{2t}{5t_0} & t < t_0 \\ \frac{2}{5} & t \geq t_0 \end{cases}$$

Multiple glues shown on the same side of a tile in Figure 3.6 are additive. For example, the glue strength between C_i and C_{i+1} ($0 \leq i \leq m-2$) is $2g_{\text{black}}(t) + g_{\text{gray}}(t)$.

This system will start assembling like a base $N^{1/j}$ counter of j digits, as briefed above and detailed in [5, 9]. It will first construct a rectangle of size $j \times N$ using $N^{1/j} + j$ type of tiles. Once the rectangle is complete, the tile on the north-west corner will start the required disassembly of the upper $(j-k)$ volatile rows, which results in the formation of a $k \times N$ rectangle. We call these two phases *Assembly phase* and *Disassembly phase* respectively, and describe them below.

Assembly Phase:

In the Assembly Phase, we aim at constructing a $j \times N$ rectangle. In the time dependent model, the assembly proceeds as in the ATA Model until the assembly of P^* tile in the k-th row (the distinguished $*$ row). At this point, an $H^{R^{**}}$ tile is required to get assembled.

However, when the $H^{R^{**}}$ tile is assembled in the $(k+1)$-th row, the total support on $H^{R^{**}}$ from its east neighbor is only $\frac{4}{5}+\frac{2}{5}<2$ at the end of μ. Thus $H^{R^{**}}$ must obtain additional support; otherwise it will get disassembled, blocking the desired assembly process. The additional support comes both from its south neighbor and its west neighbor. (1) On the south front, tile R^* can arrive and be incorporated in the k-th row (the distinguished $*$ row) of the assembly. It holds $H^{R^{**}}$ for another time interval of μ and provides a support of $\frac{2}{5}$. Further note that during this second interval, an R tile can be assembled in the $(k-1)$-th row, and the R^* tile in the k-th row will then have support 2 at μ and hence stay attached. In addition, tile R has support 2 at μ, so it will also stay attached. Regarding $H^{R^{**}}$, the end result is that it receives an additional *stable* support $\frac{2}{5}$ from its south neighbor. However, the maximum support from both the south and the east is at most $1+\frac{1}{2}+\frac{2}{5}$, which is still less than $\tau=2$. Fortunately, additional rescue comes from the west. (2) On the west front, an i^{**} tile can get attached to $H^{R^{**}}$, and stabilize it by raising its total support above 2. However, this support is insufficient, in the sense that i^{**} itself needs additional support from its own west and south neighbors to stay attached. If this support can not come in time, that is, before μ, i^{**} will get disassembled, in turn causing the disassembly of $H^{R^{**}}$. The key observation here is that this assembly/disassembly is a reversible dynamic process: the disassembly may stop and start going backwards (*i.e.* assembling again) at any point. Thus in a dynamic, reversible fashion, the target structure of the Assembly Phase, namely the $j\times N$ rectangle, can be eventually constructed.

The above added complication is due to the fact that we require the $H^{R^{**}}$ tiles in the $(k+1)$-th row to get a total support of <2 from the south and the east. This is crucial because during the subsequent Disassembly Phase (as we describe next) the desired disassembly can only carry through if the total support of each volatile tile from the south and the east is <2.

Disassembly Phase:

In the Disassembly Phase, we will remove the $j-k$ volatile rows, and reach the final target structure, a $k \times N$ rectangle. Once the $j \times N$ rectangle is complete, the tile T at the north-west corner (P' tile in the j-th row) initiates the disassembly. When the μ of the glue-pairs between tile T and its neighbors is over, tile T will get detached because the total glue strength that it has accumulated is $\frac{4}{5} + \frac{2}{5} < \tau = 2$. Note that, unlike the above case for $H^{R^{**}}$, no additional support can come from the west for tile T since T is the west-most tiles. As such, T is doomed to get disassembled. With T gone, T's east neighbor will get removed next, since it now has a total glue strength $\leq 1 + \frac{1}{2} < \tau$. Similarly, all the tiles in this row will get removed one by one, followed by the removal of the tiles in the next row (south row). Such disassembly of the tiles continues until we are left with the target rectangle of size $k \times N$, whose constituent tiles, at this stage, all have a total glue strength no less than $\tau = 2$, and hence stay stably attached.

Note that, similar as in the Assembly Phase, the volatile tiles that just got removed might come back. But again, ultimately they will have to *all* fall off (after the μ), and produce the desired $k \times N$ rectangle.

Concluding the Proof:

We can construct a $k \times N$ rectangle using $O(N^{1/j} + j)$ type of tiles (where $j > k$). As in [9], it can be reduced to $O(\frac{\log N}{\log \log N})$ by choosing $j = \frac{\log N}{\log \log N - \log \log \log N}$. □ □

3.6.2 Further tiling assemblies for interesting shapes

Thin rectangles can serve as building blocks for the construction of many other interesting shapes. One example is a square of size $N \times N$ with a large square hole of size $k \times k$ ($fork \sim N$). Under the ATA Model, the lower bound can be shown to be $\Omega(\frac{(k)^{\frac{2}{N-k}}}{N-k})$ by a lower bound argument similar to the one in [9]. Note that as $N - k$ decreases, *i.e.* the square hole in the square increases, the lower bound increases. In the case when $N - k$

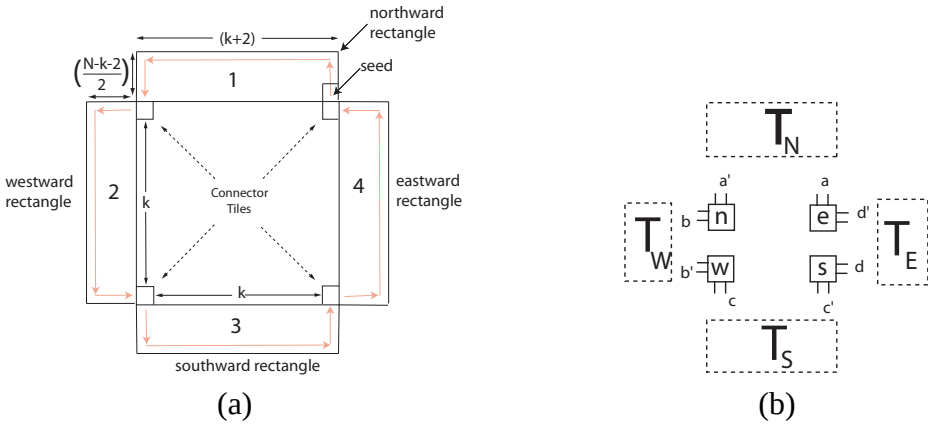


Figure 3.7: (a) Direction of the gray arrow shows the direction of construction of a square with a hole, starting from the indicated seed (b) A complete tile set for the square with hole. Sets T_N, T_S, T_W, T_E are shown in Figure 3.8, 3.9, 3.10 and 3.11

is smaller than $\frac{\log N}{\log\log N - \log\log\log N}$, the lower bound is more than $\frac{\log N}{\log\log N}$. In the case when $N-k$ is a small constant, the complexity is almost N^c, where c is some constant < 1. However, in time-dependent model, the tile complexity of this shape can be reduced to $O(\frac{\log k}{\log\log k})$ even for small values of $N-k$, using our thin rectangle construction.

The basic idea is quite simple. We sequentially grow four different thin rectangles in four different directions: one rectangle northwards, one westwards, one southwards and one eastwards. The dimensions of each of these rectangles is $(\frac{N-k-2}{2}) \times (k+2)$. They will make up the major part of the square's sides as shown in Figure 3.7 a). The required tile set consists of four different groups of tile sets: each one growing a $(\frac{N-k-2}{2}) \times (k+2)$ rectangle in one direction. Each of these rectangles can be constructed by $O(\frac{\log k}{\log\log k})$ types of tiles as discussed in the proof of Theorem 8. We refer to these groups of tile sets as T_N, T_W, T_S, and T_E (Figure 3.7 (b)). The complete details of tile sets T_N, T_W, T_S, and T_E are shown in Figures 3.8, 3.9, 3.10 and 3.11.

As shown in the center in Figure 3.7 b), we need some additional tiles (tiles n, e, s, and w) to connect these four different rectangles with each other in order to complete the desired square with a hole. We call them *connector tiles*. Note that the glues on the sides of

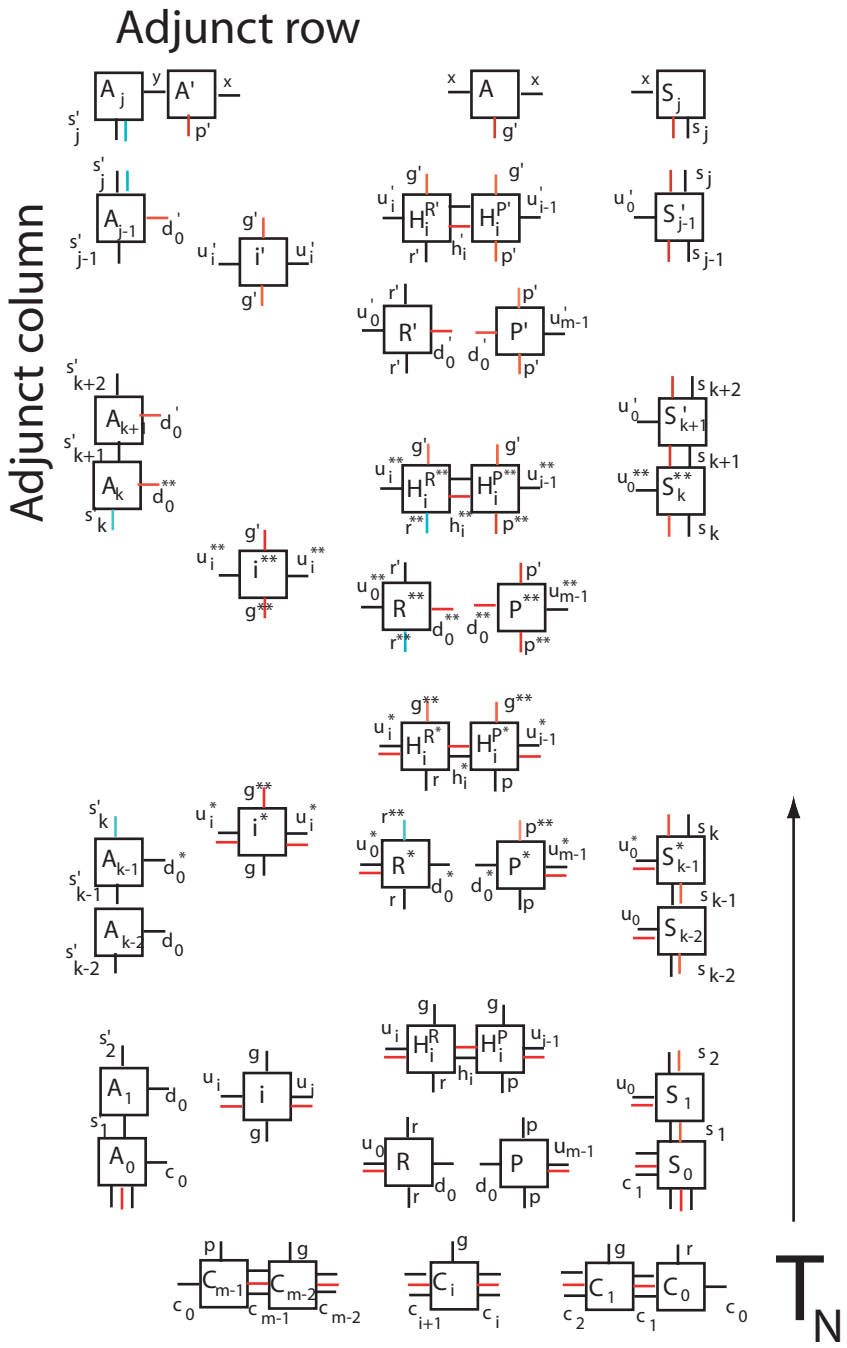


Figure 3.8: (a) Figure displays the tiles from the sets T_N required for the construction of $N \times N$ square with a hole of size $k \times k$ in the center. It should be noted that symbols in the Figures 3.8, 3.9, 3.10, and 3.11 are from different namespaces. It means that a glue-symbol x in T_N is different from a glue-symbol x in T_S, T_W or T_E, and they can not interact

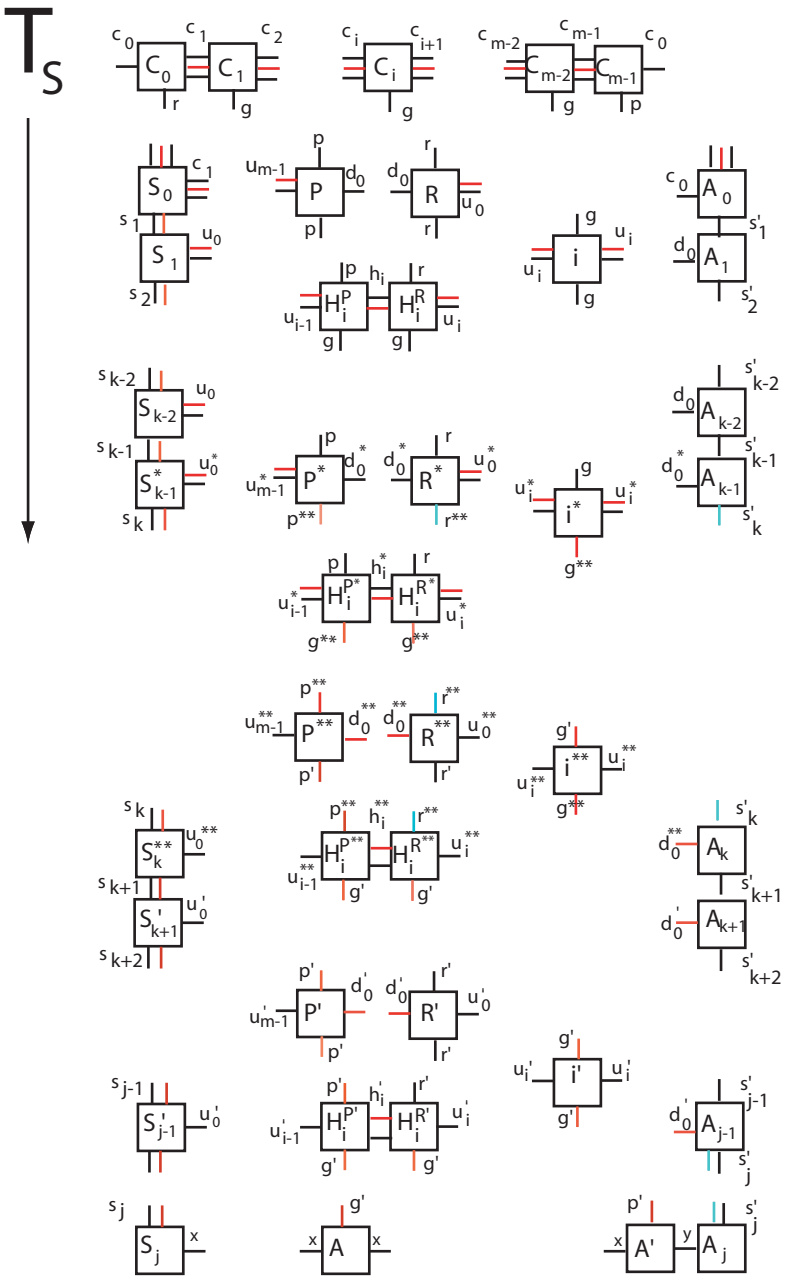


Figure 3.9: Figure displays the tiles from the sets T_S required for the construction of $N \times N$ square with a hole of size $k \times k$ in the center

connector tiles match with the glues of *seed* tiles of the corresponding rectangles. After the completion of one rectangle the corresponding connector tile should assemble and provide path for the assembly of another rectangle. For example, the assembly of the connector tile

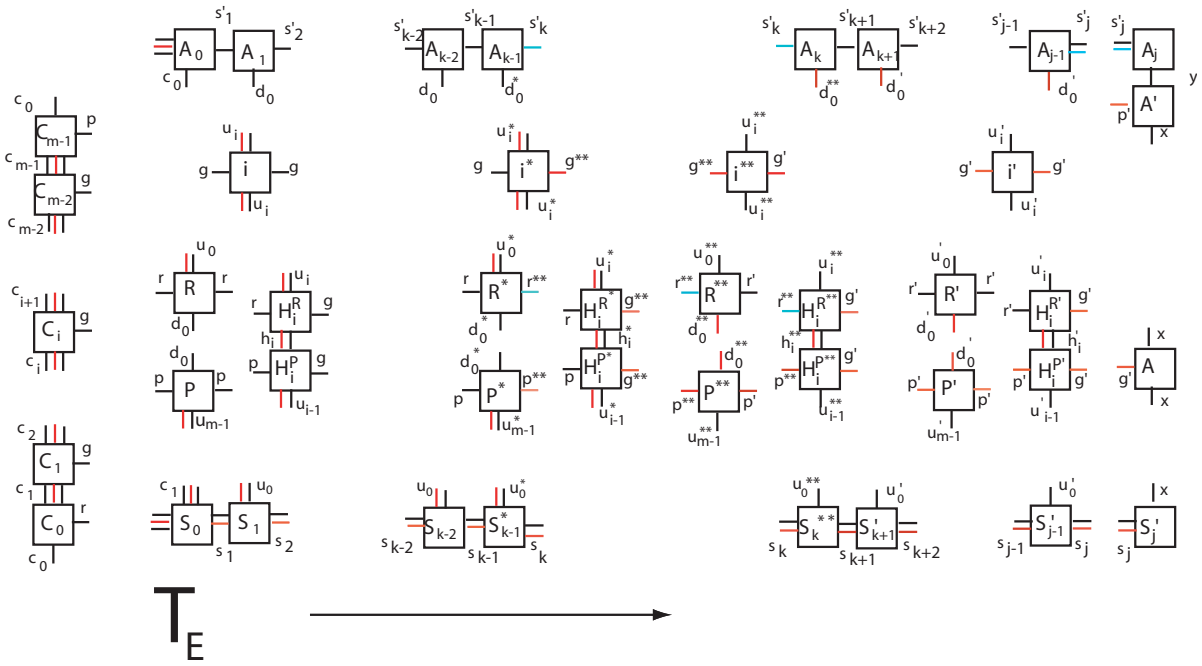


Figure 3.10: Figure displays the tiles from the sets T_E required for the construction of $N \times N$ square with a hole of size $k \times k$ in the center

n takes place after the assembly of the west-most column of the northward rectangle from the tile set T_N, and triggers the assembly of the westward rectangle.

In each of the thin rectangles, a special row and a special column is needed that can assist the assembly of the corresponding connector tile. We call these special rows and columns as *adjunct row* and *adjunct column*. The tiles required for the assembly of the adjunct row and column in the northward rectangle are shown in Figure 3.8. Note that the glues on the sides of these tiles are designed in such a way that they do not inhibit the disassembly phase in the construction of the corresponding thin rectangle.

Finally, we have gaps at the four corners this $N \times N$ square, and a $(k+2) \times (k+2)$ square hole in the center with exactly one tile present at each corner of the hole (Figure 3.7). A constant number of type of tiles, referred to as *filler tiles*, will be needed to fill in these gaps, and obtain an $N \times N$ square with a $k \times k$ hole at its center.

The complete tile set T_N is the tile set described in the proof of Theorem 8 along with

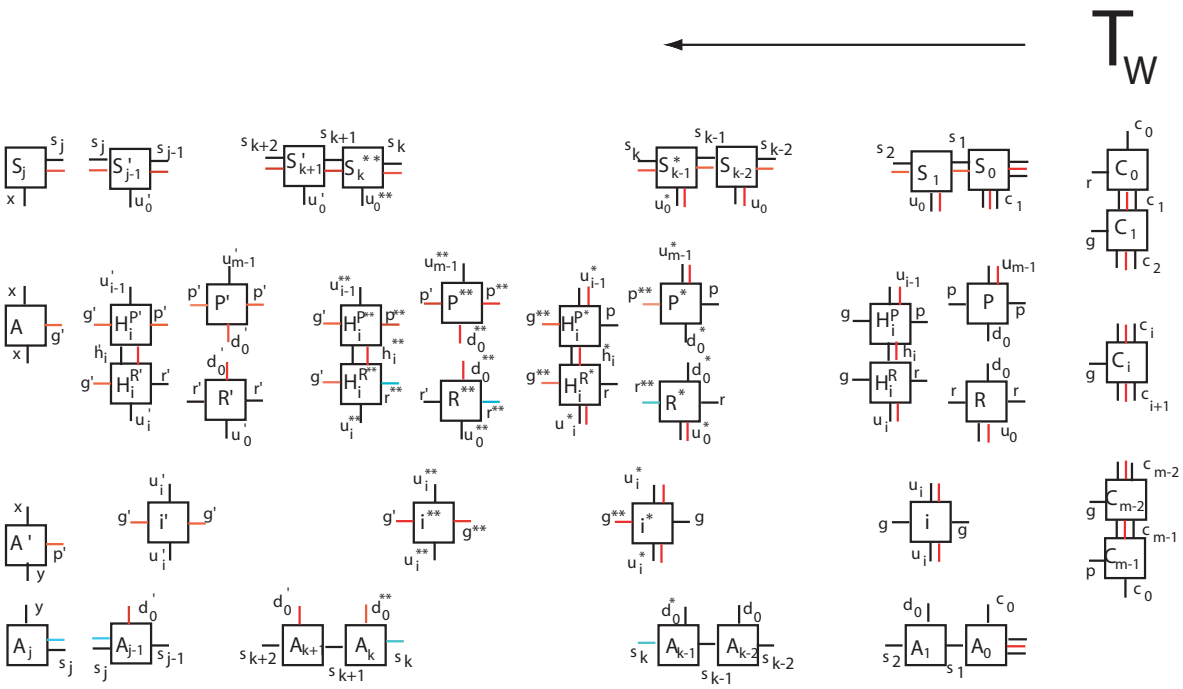


Figure 3.11: Figure displays the tiles from the set T_W required for the construction of $N \times N$ square with a hole of size $k \times k$ in the center

the tiles for adjunct row and column. The boundary tiles in T_N are modified slightly so that they can assist the assembly of appropriate filler tiles when required. Tile set T_W is formed by rotating every tile of T_N anticlockwise by $90°$. It should be noted that a totally disjoint set of symbols for the glues should be used in T_W to avoid any interaction with the tiles in T_N. Similarly, the tile sets T_S and T_E can be obtained by further anticlockwise rotation of T_W by $90°$ and $180°$ respectively.

Thus, the total number of tiles required is the sum of tiles required for each of the four thin rectangles, four connector tiles, constant number of filler tiles, and the tiles for adjunct row and column in each of the four rectangles. This is upper bounded by $O(\frac{\log k}{\log \log k})$. The assembly will grow in the manner shown in Figure 3.7 a). Assuming without loss of generality that the *seed* is the seed tile of rectangle 1. Then first rectangle 1 will be constructed; then the connector to rectangle 1 and 2 will assemble; then rectangle 2 will

assemble; then connector to rectangle 2 and 3; then rectangle 3; then connector to 3 and 4; finally rectangle 4 will get assembled. It should be noted that the filler tiles can assemble anytime during the assembly, whenever they get enough support to hold them.

3.7 Discussion and Future Work

In this chapter, we defined a model in which the glue strength between tiles depends upon the time they have been abutting each other. Under this model, we demonstrate and analyze catalysis and self-replication, and show how to construct a thin $k \times N$ rectangle using $O(\frac{\log N}{\log \log N})$ tiles for constant $k > 0$. The upper bound on assembling a thin rectangle is obtained by applying similar assembly strategy as in the multi-temperature model [9]. Thus, an interesting question is whether the multi-temperature model can be simulated using our time-dependent model. It is also an open problem if under our model the lower bound of $\Omega(\frac{\log N}{\log \log N})$ for the tile complexity of an $N \times N$ square can be further improved.

Another interesting direction is to study the kinetics of the catalysis and self-replication analytically. Winfree's kinetic model [190] can be used to study them, but the challenge here is that the rate constant for the dissociation for a particular species varies with time because of changing glue strengths of its bonds. This makes the analytical study hard. However, these catalytic and self-replicating systems can be modeled as a continuous time markov chain, and studied using computer simulation to obtain empirical results.

Chapter 4

A Framework for Modeling DNA based Molecular Systems

Recent successes in building large scale DNA nanostructures and in constructing DNA nanomechanical devices have inspired scientists to design more complex nanoscale systems. The design process can be made considerably more efficient and robust with the help of simulators that can model such systems accurately prior to their experimental implementation. In this chapter, we propose a framework for a discrete event simulator for simulating the DNA based nanorobotical systems. It has two major components: a physical model and a kinetic model. The physical model captures the conformational changes of molecules, molecular motions and molecular collisions. The kinetic model governs the modeling of various chemical reactions in a DNA nanorobotical systems including the hybridization, dehybridization and strand displacement. The feasibility of such a framework is demonstrated by some preliminary implementations.

4.1 Introduction and related work

4.1.1 Motivation

Recent research has explored DNA as a material for self-assembly of nanoscale objects [41, 94, 109, 159, 195, 201, 202], for performing computation [3, 22, 20, 21, 105, 104, 108, 189, 190, 196], and for the construction of nanomechanical devices [10, 45, 46, 58, 102, 110, 175, 133, 160, 161, 162, 163, 179, 178, 203, 209, 210]. One potential application of an autonomous unidirectional DNA device is to perform computation. Recently Yin et al [207, 205] proposed the design of an autonomous universal turing machine and

cellular automata. Another potential application beyond computation is the design of a controllable moving device integrated into a DNA lattice for efficient transportation. The major challenges in front of the researchers interested in designing complex DNA based nanodevices, are the time consuming and costly experiments. A lot of times the effect of alterations in only a few parameters need to be tested, and the entire set of experiments need to be repeated from scratch. Accurate computer simulations that capture the essential physical and chemical properties can serve as an effective tool in the design process.

4.1.2 Prior Simulators for DNA Computing

Previous simulators for DNA computing include:

- *VNA simulator*[121, 122]: a simulator to aid the protocols for DNA computing. The simulator consists of two main parts, one for finding reactions among existing molecules and generating new ones, and the other for numerically solving differential equations to calculate the concentration of each molecule.

- *Virtual test tubes*[65, 66, 67]: a simulator for biochemical reactions based on the kinetics of molecular interactions.

  Neither of these deal with the shapes of nanostructure, and therefore not suitable for simulation of nanorobotics or nanofabrication applications.

- *Hybrisim*[77]: a simulator that deals with the detailed simulation of hybridization only between two strands, and therefore, very limited in use.

Sales-Pardo et. al. [150] modeled a ssDNA as a bead-pin rotational polymer chain and used a modified Monte Carlo simulation to investigate the dynamics of a single-stranded DNA and its associated hybridization events. The geometric constraints of the nucleic chain were handled by a lattice model.

Isambert and Siggia [78] modeled RNA helices as rods and single stranded RNA as Gaussian chains. Kinetic Monte Carlo method was used to sample RNA conformational changes. They also used the short-scale and the large-scale conformation descriptors, i.e. *nets* and *crosslinked gel*, to model geometric constraints related to complex RNA folding conformations.

Bois et. al. [27] investigated the possible effects of topological constraints in DNA hybridization kinetics. Recently Dirks et. al. [55] developed an algorithm aimed at analyzing the thermodynamics of unpseudo-knotted multiple interacting DNA strands in a dilute solution.

4.1.3 Our Results and Organization of this Chapter

In this chapter, we describe a comprehensive framework for simulation of DNA based nanorobotic devices.

Our method of simulation is different from the commonly used Gillespi algorithm [69, 87, 70, 180, 62, 130]. In the Gillespi algorithm the concentrations of various reactants are stored as $X_1, \ldots X_n$. And also the rate constants of various possible chemical reactions are also stored as $c_1, \ldots c_m$. Then the calculation of rates of various reactions gives the probabilities for various reactions. The appropriate reaction R_μ is then chosen probabilistically, and after the execution of this reaction the concentrations $X'_1, \ldots X'_n$ of various chemicals are updated appropriately. The algorithm is computationally expensive: more so, in the systems of our interest where the number of macromolecular interactions are too large. For example, the potentially huge number of possible products from two different DNA single stranded molecules depending upon their alignment with each other, and each product formation will have a different rate constant. Moreover, in nanorobotical and nanofabrication applications, the shapes of various nanostructures involved are as important as the concentrations of the reactants and the reaction rates. Therefore, phys-

ical simulations are performed to model the molecular conformations and the chemical reactions are monitored explicitly.

In this chapter, we describe a framework for the design of a discrete event simulator, which simulates DNA based nanorobotical devices. Section 4.2 gives an overview of the system. Section 4.3 describes the physical simulation of the molecules. Section 4.4 discusses the event simulation based on the kinetic and thermodynamic studies. Section 4.5 describes the adaptive time-steps to optimize the physical simulation, and Section 4.6 describes the analysis of the complete algorithm. Section 4.7 presents some preliminary results to support such a framework. Discussions and future work is described in Section 4.8. It should be noted that in this chapter, we present the framework for building such a simulator and not the simulator itself. In the subsequent text any reference to simulator is a reference to this framework.

4.2 Our Discrete Event Simulation

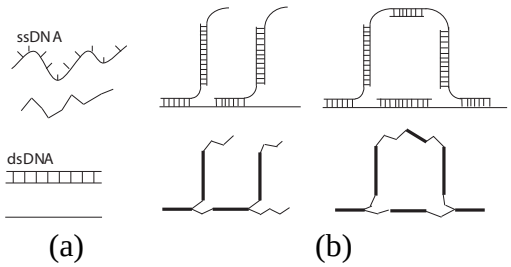


Figure 4.1: (a) Schematic view of the molecules in the modeled system. Bold solid lines represent the worm-like chain (WLC) model used for dsDNA segments while thin solid lines represent the WLC model used for ssDNA segments. (b) Figure shows a complex DNA nanostructure reduced to a collection of WLC segments with different parameters (bold solid line for dsDNA and thin line for ssDNA segments)

The simulator performs the molecular-level simulations and provides an useful tool to study DNA based nanomechanical devices. It has two major components. The first component is the physical simulation of the molecule conformations. The second component is

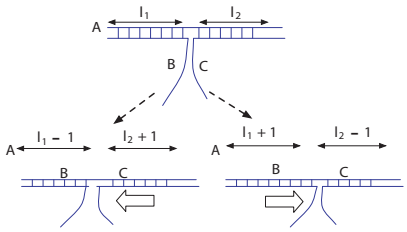


Figure 4.2: Strand displacement: molecule B and C compete against each other to hybridize with molecule A

the event simulation (hybridization, dehybridization and strand displacement events) which depends on the kinetic and thermodynamic properties of the molecules. Due to the large number of molecules in a given solution, we sample and simulate molecules within a small test volume, assuming the solution is uniform.

The modeled system consists of three types of molecules, single-stranded DNA (ssDNA) molecules, double-stranded DNA (dsDNA) molecules and complex DNA nanostructures with both single-stranded and double stranded segments, as shown in Figure 4.1.

For the sake of simplicity, we assume no pseudo-knots formation for the complex DNA nanostructures. Therefore, to a first approximation, the complex DNA nanostructure is reducible to a collection of WLC segments with different parameters (i.e. persistence length, elasticity, diffusion coefficients etc). For more complicated DNA nanostructures, we can adapt the geometric descriptors used in [78, 27], as discussed in Section 4.8.

The secondary structure of a nanostructure can be represented as an undirected graph called connectivity graph. Individual strands are represented as nodes, and hybridization relationships between strands are represented by edges between corresponding nodes. There is an edge between the nodes representing two strands, if and only if the two strands are hybridized with each other. An appropriate data structure for this will be an adjacency list. However, with every node (i.e. double stranded region of any strand) the information about its neighboring unhybridized region also needs to be stored. As shown in Figure 4.2, this data structure stores individual molecular configurations including sequence and sec-

ondary structure.

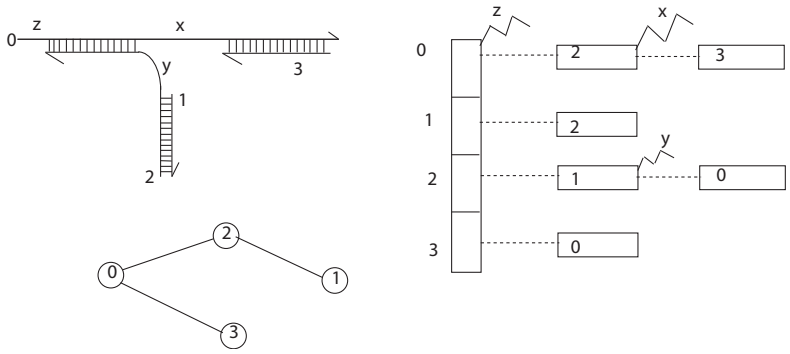


Figure 4.3: Suggested data structure for modeling the DNA based complex nanostructures, and connectivity graph for the strands in the nanostructure. It should be noted that the information about the unhybridized sections of the strands is stored at the nodes that represent the neighboring duplex portions as shown in the Figure

During the simulation, three types of reaction take place in the solution: the hybridization between a pair of ssDNA segments with complementary base-pairing, the dehybridization of the dsDNA portion of a nanostructure and the strand displacement. The DNA molecules contain potential hybridization sites at their free-ends (sticky ends). During the simulation, when two molecule come into contact (reactive collision), a potential hybridization event is reported. The corresponding free-end base-pairs are investigated to determine the probability of its actual occurrence. Strand displacement is a reaction in which two strands compete against each other to hybridize with a common strand as shown in Figure 4.2. Strand B and C compete against each other to hybridize with strand A. At a time instance, B (or C) makes one more bond with A and removes one bond of C (or B).

The required discrete event simulation with Δt as the time-interval is described as follows. Algorithm 1 describes the major steps of the simulation. MQ stores all the nanostructures in the system. T is the total simulation time. Δt is the simulation time per step. *Initialize* is a function that initializes the MQ based on the user input. The detailed algorithms are described in the subsequent Sections.

Algorithm 2 describes steps involved in generating random conformations for all molecules

in the system. *Enqueue* and *Dequeue* are standard queuing operations that insert and delete an element in the queue. *MCSimulation(m)* generates new conformations for the molecule m.

Algorithm 1 Discrete Event Simulation

Initialize(MQ)
while $t \leq \mathsf{T}$ **do**
  $t = t + \Delta t$
  {PHYSICAL SIMULATION}
  Physical simulation
  Collision detection
  {EVENT SIMULATION}
  Hybridization
  Dehybridization
  Strand displacement
end while

Algorithm 2 Physical Simulation

for $\forall m_i \in MQ$ **do**
  MCSimulation(m_i)
end for

Algorithm 3 MCSimulation (m)

m^* =RandomConformation(m)
if SelfCollision(m^*) **then**
  continue to next iteration
end if
$\Delta E = E(m^*) - E(m)$
if ($\Delta E > 0$) **then**
  $x \in_{var} [0, 1]$
  if ($x > \exp -\frac{\Delta E}{K_B T}$) **then**
    continue to next iteration
  end if
end if
$m = m^*$

Algorithm 4 Collision Detection

for $\forall m_i, m_j \in MQ, i \neq j$ **do**
  if collide(m_i, m_j) **then**
    e =HEvent(m_i, m_j)
    Enqueue(HQ, e)
  end if
end for

Algorithm 5 Hybridization

while HQ is NOT empty **do**
  $e =$ Dequeue(HQ)
  Hybridize(e)
  Update(MQ)
  if PotentialSD(e) **then**
    Enqueue(SDQ, e)
  end if
end while

Algorithm 6 Dehybridization

for $\forall m_i \in MQ$ **do**
  for $\forall b \in$ bonds of m_i **do**
    if PotentialDehybridization(b) **then**
      Dehybridize(b)
    end if
  end for
  if any dehybridization performed **then**
    DFS on connectivity graph of new m_i, each connected component is a new molecule formed.
    Update(MQ)
  end if
end for

Algorithm 7 Strand Displacement

while SDQ is NOT empty **do**
  $e =$ Dequeue(SDQ)
  $e^* =$ StrandDisplacement$(e, \Delta t)$
  if IncompleteSD(e^*) **then**
    Enqueue(SDQ^*, e^*)
  end if
  Update(MQ, e^*)
end while
$SDQ = SDQ^*$

The *MCSimulation(m)* function, described as Algorithm 3, is based on the Metropolis algorithm[73, 117]. $E(m)$ is the energy associated with conformation of molecule m. ΔE, defined as the energy change of the system due to the transition to new conformation, determines the probability that the molecule achieves the new conformation.

Algorithm 4 describes reactive collision detection which leads to potential hybridization events. *Collide(*m_i, m_j*)* returns true if the sticky ends of molecule m_i and m_j collide. e is a data structure that stores an event (hybridization, dehybridization or strand displacement), including all the molecular configurations involved in the event and supplementary information related to the event. For example, in the case of hybridization, it stores the molecular configurations and the information of the hybridization sites. *HEvent(*m_i, m_j*)* creates a potential hybridization event based on a collision between molecules m_i and m_j. HQ stores all potential hybridization events.

Algorithm 5 presents the algorithm involved in hybridization. *Hybridize(e)* probabilistically determines the hybridization product based on the change in free energy as described in Section 4.4. *PotentialSD(e)* returns true if event e is a potential strand-displacement event. SDQ stores all potential strand-displacement events. When two nanostructures hybridize to form a larger nanostructure, their corresponding connectivity graphs are merged together to form the connectivity graph representing the newly formed nanostructure. *Update(MQ)* updates the configurations of the molecule in the system based on the occurred event e.

Algorithm 6 describes the dehybridization event. *PotentialDehybridization(m)* returns true if molecule m could potentially dehybridize. *Dehybridization(m)* probabilistically dehybridizes molecule m. The corresponding edges are deleted from the connectivity graph representing the nanostructure. Each connected component, thus formed, represents one newly formed nanostructure. *Update(MQ)* updates the configurations of the molecules in the system based on the event e that occurred.

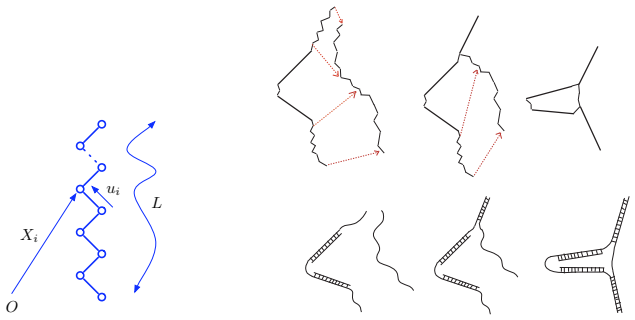
Algorithm 7 shows the steps involved in the strand displacement event. *StrandDisplacement*($e, \Delta t$) probabilistically proceeds with the strand displacement event e within time frame Δt. *IncompleteSD(e)* returns true if the strand displacement event has not completed within the given time frame.

4.3 Our Physical simulation

The discrete worm-like chain model (WLC) is used to model the polymer-like DNA molecules in solution. Monte Carlo (MC) computer simulations are used to determine their conformations.

4.3.1 The Discrete Wormlike Chain Model for DNA

The advancements in the experimental study of single molecule dynamics offers opportunities for experimental validations of various DNA polymer models, among which Gaussian Chain Model, Freely-Jointed Chain (FJC) and Worm-Like Chain (WLC) are widely investigated [127, 92, 80, 165, 123, 59, 91, 13, 199, 166, 95, 31, 36, 93]. The choice of a polymer model depends on the physical property of the DNA chain, affordable computation and molecular-details of interest [56]. Our simulation is constructed using the



(a) (b)

Figure 4.4: (a) WLC model (b) Figure illustrates various steps with respect to the physical motion of the strands during hybridization

discrete wormlike chain model. Marko and Siggia [111, 112] used the model to derive the elastic theory suitable for DNA and further completed the model to include bending and twisting elasticity of DNA and the free energy required for deformation. Bustamante et al [36] proposed an interpolation of the Marko-Siggia model for fitting and experimental elasticity curve of single DNA molecules. Klenin et al [90] modeled linear and circular DNA where the DNA polymers are represented by a WLC of stiff segments connected by bending torsion and stretching potentials. Tinnoco et al [177] used WLC as their polymer chain conformation to investigate force effect on thermodynamics and kinetics of single molecule reaction. Larson et al [99, 54] used a similar model to predict the behavior of tethered dsDNA in a constant-velocity flow. Experimental data has shown some reasonably good agreement with the model [118].

The DNA molecule (Figure 4.4 (a)) is initialized as $N+1$ beads $(0, 1..N)$ connected by N mass-less extendable segments (springs) of the same length [54, 61, 100]. The contour length of the chain is L. The position of the bead i is denoted as $\mathbf{x}_i$. The segment vectors are given by

$$\mathbf{u}_i = \mathbf{x}_i - \mathbf{x}_{i-1} \tag{4.1}$$

Therefore the chain is represented by a set of $N + 1$ vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ [40]. We use WLC to model ssDNA, dsDNA and complex DNA nanostructures. Specifically for a complex DNA nanostructure, different parameters as described in Section 4.3.5 are applied to different segments of the chain depending on whether the segment is double-stranded or single-stranded.

4.3.2 Monte Carlo Simulation

The molecules are simulated through Monte Carlo simulation for a desired number of time steps using Algorithm 3. According to the Metropolis algorithm used in the simulation, $E(m)$ is the energy associated with conformation of molecule m. The computation of

$E(m)$ will be discussed in Section 4.3.4. ΔE is defined as the energy change of the system due to the new conformation. K_B is the Boltzman constant, and T is the absolute temperature. MQ is the set of all molecules in the simulation. *RandomConformation* is a function that achieves a new conformation of the molecule through random walk in three dimension. *SelfCollision* detects and excludes the self-crossing conformations. The detailed algorithm is shown in Algorithm 3. Similar methods have been used in [211, 15, 107].

4.3.3 Random Conformation

The random conformation of the DNA molecule is generated by a random walk in three dimensions. Based on [14],

$$\Delta \mathbf{x}_i = \mathbf{R}_i \tag{4.2}$$

where $\Delta \mathbf{x}_i$ is the change of $\mathbf{x}_i$ in time step Δt, and $\mathbf{R}_i$ is the random displacement. Let D be the diffusion coefficient. We assume $\mathbf{R}_i$ as a Gaussian random variable which is distributed according to

$$W(\mathbf{R}_i) = (4A\pi)^{-3/2} \exp(-\mathbf{R}_i/4A) \tag{4.3}$$

where $A = D\Delta t$. The diffusion coefficient D of a macromolecule in an ideal dilute solution is computed according to $D = K_B T/f$, where f is the hydrodynamic frictional coefficient of the macromolecule [171]. For a rigid, rod-like molecule f can be written as $f = 3\pi\eta L/(\ln \rho + \gamma)$, where η is the viscosity of the solution, L is the length of the DNA molecule, ρ is the axial ratio and γ is a correction for end effects [171].

4.3.4 Energy

Now we describe how we calculate $E(m)$ as stated in Algorithm 3. Our current simplified model neglects the following energies (though a more accurate model should take them into consideration [211, 53]): pairing potential between complementary bases, stacking

energy from the vertical interactions between neighboring base pairs and hydrodynamic interaction energy with the solvent. We shall consider the torsional rigidity in the forms of bending torque and twisting torque for the DNA molecules in a more sophisticated model. The total energy of a DNA conformation is given as the sum of stretching, bending, twisting and electrostatic interaction energy among negatively charged phosphate groups along the chain [90, 211, 97], which are denoted as E^s, E^b, E^t and E^e, respectively.

$$E^{total} = E^s + E^b + E^t + E^e \tag{4.4}$$

Stretching Energy. The stretching energy is defined as

$$E^s = \frac{1}{2} Y \sum_{i=1}^{N} (u_i - l_0)^2 \tag{4.5}$$

where l_0 is the segment equilibrium length, Y is the stiffness parameter defined previously [211].

Bending Energy. The bending elastic energy of the coarse-gained bead-rod model is

$$E^b = -\frac{\kappa}{l_0} \sum_{j=2}^{N} \frac{\mathbf{u}_j \cdot \mathbf{u}_{j-1}}{|\mathbf{u}_j||\mathbf{u}_{j-1}|} \tag{4.6}$$

where l_0 is the length of the connecting rod and $\mathbf{u}_j/|\mathbf{u}_j|$ is the unit vector directed from bead j-1 to j [56, 181]. The bending rigidity κ is related to the persistence length P by

$$\kappa = K_B T P \tag{4.7}$$

Twisting Energy. The twisting energy is defined as

$$E^t = \frac{C}{2l_0} \sum_{i=2}^{N-1} (\tau_i)^2 \tag{4.8}$$

where l_0 is the segment equilibrium length, C is the torsional rigidity constant and τ_i is the twist angle between the $(i-1)$th and ith segments [90]. The computation of the twist energy can be found in [90].

Electrostatic Energy. The DNA intra-chain electrostatic repulsion/attraction can be described by the Debye-Hükel approximation as the electrostatic potential between any two non-adjacent segments i and j [211, 97, 90],

$$E^e_{i,j} = \frac{\nu^2}{D} \int d\lambda_i \int d\lambda_j \frac{\exp\left(-\kappa r_{i,j}\right)}{r_{i,j}} \tag{4.9}$$

where $r_{i,j}$ is the distance between two charges at arc length parameters $d\lambda_i$ and $d\lambda_j$ along the chain. κ is the inverse of the Debye length and is given as $\kappa = 8\pi e^2 I/K_B T D$, where I is the ionic strength, e is the proton charge, and D is the dielectric constant of water. ν is the linear charge density, which is $\nu = -2e/\Delta$, where Δ is the distance between base pairs [90, 97].

4.3.5 Parameters

We use the WLC model for both ssDNA and dsDNA for modeling consistency. It is important to notice that there are different sets of parameters used for each of them.

Parameters for ssDNA. Let L be the contour length of the ssDNA, $L = l_{bp} N_{bp} = l_0 N$. Here l_{bp} is the length of the ssDNA per base pair. N_{bp} is the number of bases. N is the number of beads (monomer) in our WLC model. l_0 is the length per segment. The average length of ssDNA in the system is approximately $25 - 30\ bp$. According to [204], $l_{bp} = 0.7\ nm$. Many groups have obtained the force/extension data for ssDNA in different salt environments[211, 165, 140, 106, 37]. Parameters used in our model are obtained from [211], where $l_0 = 1.5\ nm$ and $Y = 120\ K_B T/nm^2$. The persistence length P is $0.7\ nm$ [165]. The diffusion coefficient D of ssDNA is obtained from [171] as approximately $1.52 \times 10^{-6}\ cm^2 s^{-1}$ for a $20\ bp$ strand. The diameter of the ssDNA backbone is $1\ nm$ [52].

Parameters for dsDNA. For dsDNA, the parameters associated with the equations are different, i.e. $l_0 = 100\ nm$ [90, 50, 115], $P = 50\ nm$, $Y = 3K_BT/2P$ [50, 172], $l_{bp} = 0.34\ nm$ [204], and $D = 1.07 \times 10^{-6}\ cm^2 s^{-1}$ [171]. For a short dsDNA segment ($20\ bp$), the WLC model can be simplified to the straight, rigid cylinder model with reasonable adequacy [11, 115]. WLC models are used for simulation consistency.

4.3.6 Motion of the complex nanostructure.

The MC simulation described previously can be applied to a complex nanostructure. Such a nanostructure is reducible to a collection of ssDNA and dsDNA WLC segments as shown in Figure 4.1 b). Perturbations to each segment are done independently. The total energy is computed as a summation of the energies associated with individual segments. For a more accurate model, loop energy [26, 19] of DNA strands can also be considered in the DNA nanostructures that contain the loops in the systems of our interest.

4.3.7 Physical model for hybridization

Though extensive research has been done for RNA folding simulation [60, 197], to the best of our knowledge there is no empirical results that describe:

1. The motions of each individual strand during the hybridizations.
2. The actual physical location of the hybridized products relative to other molecules in the system.

Therefore we make the following hypotheses:

1. Upon collision that leads to potential hybridization, two strands immediately align their bases involved in the formation of duplex with the right orientation.
2. During the hybridization process, the displacement of the two strands is inversely proportional to their masses (or number of bases in the structure).

The model can be subsequently improved as the empirical evidence becomes available. Figure 4.4 b) illustrates one schematic to depict our hypotheses.

4.3.8 Discussions of other physical models

To calculate the mesoscale dynamics of the DNA molecules, Brownian Dynamics (BD) simulation techniques can be applied to replace explicit solvent molecules with a stochastic force [56, 97, 90, 81, 74, 38, 169, 98, 76]. To obtain more accurate molecular details of a particular DNA nanostructure, we may improve our MC physical model with the BD simulation. However, BD simulation becomes infeasible as the number of molecules increases per test volume. For descriptions of the BD algorithm and explicit force computations, refer to [12, 90, 116, 97].

4.4 Event Simulation

In the event simulation module we use thermodynamics and kinetics principles to calculate the probabilities of various events. Possible events in systems of our interest are hybridization, dehybridization (melting/dissociation) and strand displacement.

4.4.1 Hybridization

The nearest-neighbor (NN) model is used to model the hybridization event[83]. The model assumes that the stability of a given base-pair depends on the identity and orientation of neighboring base pairs [83]. Empirical data is used to determine parameters for all possible alignments of base pairs. The model has been shown to describe the thermodynamics of DNA structures that involve mismatches and neighboring base pairs beyond the Watson-Crick pairs [129, 151].

Let ΔG° be the standard free energy released as heat by a single hybridization event.

It can be calculated from the standard enthalpy and entropy of the reaction [83]:

$$\Delta G^\circ = \Delta H^\circ - T\Delta S^\circ \tag{4.10}$$

For reactions taking place in commonly used buffers, the standard enthalpy and entropy can be reliably estimated from the oligonucleotide sequence according to a nearest neighbor stacking model [72]:

$$\Delta H^\circ = \Delta H^\circ_{ends} + \Delta H^\circ_{init} + \sum_{k\in\{stacks\}} \Delta H^\circ_k \tag{4.11}$$

$$\Delta S^\circ = \Delta S^\circ_{ends} + \Delta S^\circ_{init} + \sum_{k\in\{stacks\}} \Delta S^\circ_k \tag{4.12}$$

A coarser approximation for DNA of length s can be used[190], so that $\Delta H^\circ \sim -8s\ kcal\ mol^{-1}$ and $\Delta S^\circ \sim (-22s-6)\ cal\ mol^{-1}\ K^{-1}$. *BIND* [72], the thermodynamic simulator for DNA hybridization can be used to calculate the ΔH°, ΔS° and ΔG° for the reaction between two DNA molecules.

When a potential hybridization event that involves molecules m_1 and m_2 is detected due to a collision, the simulator examines all possible alignments of m_1 and m_2. For hybridization according to alignment i, its free energy ΔG°_i is computed using the NN model. Let $m_1{m_2}^i$ be its hybridization product. Let p_i be the stability measurement of $m_1{m_2}^i$. Then it is known that $p_i \propto \exp(-\Delta G^\circ_i/RT)$. Let P^i_h be the probability of hybridization with alignment i. For all j such that p_j is below a given threshold, we reset $p_j = 0$, and only retain p_j values above that threshold. The hybridization product $m_1{m_2}^i$ is formed with probability P^i_h, where

$$P^i_h = \frac{p_i}{\sum_j p_j} \tag{4.13}$$

This is represented by the formation of the connectivity graph of $m_1{m_2}^i$ by joining the

individual connectivity graphs of the molecules m_1 and m_2.

4.4.2 Dehybridization

Let k_f be the forward reaction rate constant for the hybridization and k_r be the reverse reaction rate constant (rate constant for the dehybridization). For very short DNA, the forward reaction has a diffusion-controlled rate-determining step approximately independent of its length and sequence, so

$$k_f = A_f e^{-E_f/RT} \tag{4.14}$$

where $k_f \approx 6 \times 10^5\ mol^{-1}s^{-1}$ and $A_f = 5 \times 10^8\ mol^{-1}s^{-1}$. The activation energy for the event is $E_f = 4\ kcal\ mol^{-1}$. A more accurate model can consider the effect of the DNA length, the sequence and the salt concentration on k_f [186], which is shown as,

$$k_f = \frac{k'_N\sqrt{L_s}}{N} \tag{4.15}$$

where L_s is the length of the shortest strand participating in duplex formation, N is the total number of base pairs present in non-repeating sequence, k'_N is the nucleation rate constant. For $0.2 \leq [\text{Na}^+] \leq 4.0$, k'_N is estimated as $\{4.35 \log_{10}[\text{Na}^+] + 3.5\} \times 10^5$.

The reverse reaction rate k_r is very sensitive to the DNA length and sequence:

$$k_r = k_f e^{\Delta G^\circ/RT} \tag{4.16}$$

where ΔG is the change in free energy during the hybridization (forward reaction).

Consider a molecule m_1m_2 with concentration $[m_1m_2]$. Let k_r be the rate constant for dehybridization of m_1m_2. Assuming that R_r is the reaction rate of dehybridization, we have

$$R_r = k_r[m_1m_2] \tag{4.17}$$

Thus, the number of molecules dehybridized in time Δt is $R_t\Delta t$. Therefore the probability

P_d that the molecule $m_1 m_2$ dehybridizes in Δt can be approximated as

$$P_d = \frac{k_r [m_1 m_2] \Delta t}{[m_1 m_2]} = k_r \Delta t \tag{4.18}$$

Thus, the probability of dehybridization of a double stranded section of a nanostructure is dependent on the value of k_r for that section of nanostructure.

For every molecule m_i in the system, the probability of dehybridization is evaluated for each of double stranded sections in it, and the dehybridizations of these sections is carried out probabilistically. In terms of connectivity graphs, it means the deletion of edges corresponding to the double stranded sections that dehybridized. In case more than one sections were dehybridized, *depth first search* is performed on new connectivity graph of m_i to identify individual connected components that represent the connectivity graphs of the products of dehybridization event on molecule m_i.

4.4.3 Strand Displacement

Strand displacement is modeled as a random walk in which the direction of migration of the branching point (junction) along the DNA is chosen probabilistically and is independent of its previous movements.

It has been shown that strand displacement is a biased random walk in case of mismatches [25]. In other words, migration probability towards the direction with mismatches is substantially decreased. Consider the DNA nanostructure involving molecule A, B and C in Figure 4.1(c) (top part). We refer to it as molecule ABC, and the nanostructures after 1 base pair left migration and 1 base pair right migration are referred to as $lABC$ and $rABC$, respectively. Let G°_{ABC}, G°_{rABC} and G°_{lABC} be the free energies of the molecules ABC, $lABC$ and $rABC$, respectively. Let $\Delta G^\circ_r = G^\circ_{rABC} - G^\circ_{ABC}$ and $\Delta G^\circ_l = G^\circ_{lABC} - G^\circ_{ABC}$. Let p_r be the probability of the right-directional migration and p_l be the probability of the left-directional migration. It has been shown in [25]

that $p_r \propto \exp(-\Delta G_r^\circ/RT)$, similarly $p_l \propto \exp(-\Delta G_l^\circ/RT)$, where the change of free energies can be computed by the NN model[83].

Let τ be the migration (strand displacement) time per base pair. Let N be the number of nucleotide pair migrations during a time frame of Δt. Let κ be the migration rate constant, which is the number of base pair migrated per second, $\kappa = \mathsf{N}/\Delta t$. Therefore $\tau = 1/\kappa$. At 37°, $\kappa = 6 \pm 2\ Kbp\ sec^{-1}$ and $\tau = 170 \pm 50 \mu sec$ [174]. The dependence of κ on salt concentration is discussed in [124].

Thus, the strand displacement event can be modeled as a random walk with each time step equal to τ, and probability of migration in either direction calculated as described above.

4.5 Optimizations in time stepping

The simulation captures various processes that takes place at different time-scales. Ideally, the smallest time unit should be chosen as the time step ($\delta t \sim 1\mu$ sec) to resolve the conformations and trajectory of each individual molecule using the WLC model and MC simulation. But this would make the overall simulation extremely slow. We attempt to overcome the limitations of such a short time-scale approach. Inspired by ideas in the kinetic Monte Carlo method [182], long-time system dynamics of the system consists of diffusive jumps from state to state. In general, there can be series of simulation steps where no collisions take place between the strands as they remain far apart. In the case a DNA strand is reasonably far apart (say 10-15 times its length) from all other molecules, it is treated as a rigid body, conformational changes within it can be ignored, and only its movements as a rigid body need to be considered.

Another important optimization is to increase the length of the time step itself. If all the strands are far apart, we can guarantee that within a particular time-interval δT, there will not be any collisions. In that case, a large time step δT can be taken by the simulation

to evaluate the next state of the system. When the distance reaches a given threshold where the changes in the conformations of strands can no longer being ignored, we change to the original smaller scale time step δt. The implementation of this computational efficient technique of adaptive time step, requires the distance between the closest pair of potential reactive molecules be stored and updated appropriately.

4.6 Algorithm analysis

We present an average case analysis of the simulation algorithm presented earlier. Consider that the system consists of m nanostructures each consisting of n distinct sections (single-stranded and double-stranded) when decomposed into the WLC model. For a WLC simulation of a nanostructure, since each nanostructure consists of n segments, in every run of the MCSimulation loop, the time taken is $O(n)$. Assume that on an average, the MC-Simulation loop needs to run $f(n)$ times before finding a good configuration. Therefore, the time for each step of physical simulation is $O(mnf(n))$. Naive implementation of collision detection takes $O(m^2n^2)$ time. In the event simulation part, assume that the number of collisions detected is c. Since for each collision all the alignments between two reacting strands are tested, if the average length of each single-stranded section in a molecule is l, it takes $O(cl)$. Each double stranded section is tested for a possibility of dehybridization reaction. If the average number of double stranded regions per molecule is b, then it takes $O(bm)$. For every dehybridization event, *DFS* is performed to evaluate new connected components in $O(b^2m)$. Thus, combining the physical and event simulation the total time taken in each step is $O(m^2n^2 + mnf(n) + cl + b^2m)$. The dominating terms are the first two and therefore, it can be reduced to $O(m^2n^2 + mnf(n))$. It can be concluded that the major portion of the time taken by the algorithm is in the physical simulation. Thus, it is important to optimize the time-complexity of the physical simulation of the molecules in the system as described in previous section. $f(n)$ can be extremely large, in cases where

the molecule is stuck in a low-energy conformation. Better collision detection methods can be used to improve the first term. For the strands that are a reasonable distance away from other strands, the strand can be treated as a rigid unit and the term $f(n)$ disappears causing great reduction in time-complexity. In case, all strands are far apart from each other, we may even take the advantage of using larger time steps, and fast forwarding the simulation through the uninteresting states of the system.

4.7 Preliminary Results

Our preliminary results demonstrate the feasibility of such a framework in modeling DNA based molecular systems.

4.7.1 Physical Simulation

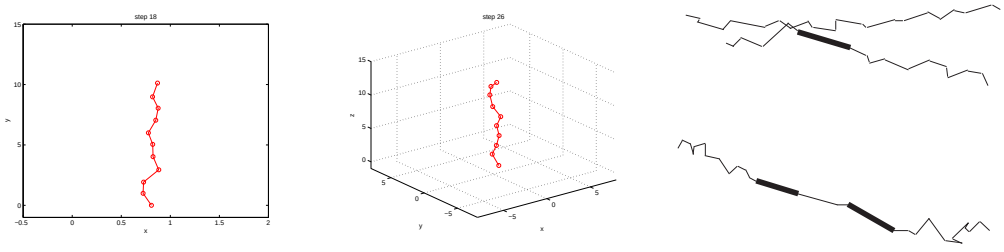


Figure 4.5: (a) 2D and 3D snapshots of the simulation for a single tethered DNA (b) Simulation of a hybridization event

The results presented here are obtained using the less computer-intensive Monte Carlo simulation of a discrete WLC model. The physical simulation module is demonstrated through the simulation of a tethered ssDNA. The same module applies to the modeling of other DNA molecules in the system. For demonstration purposes, we neglect twisting energy and focus primarily on the stretching energy and optional bending energy of the tethered DNA. Ideally, relatively long runs are carried out to generate initial conditions for simulations of the tethered-DNA chains, allowing the chains to reach their equilibrium

configurations [100]. These configurations are then saved for the actual simulation. The figures shown here are snapshots of a simulation during different time steps, from both 2D and 3D perspective (Figure 4.7.1 (a)). The scales for the x-axis and the y-axis are enlarged to show the details of the conformational changes relative to the horizontal plane. The simulations are preliminary but promising.

4.7.2 Event Simulation

We present here a snapshot of a hybridization event in simulation based on our framework in Figure 4.7.1 (b). Bold black lines represent the double stranded DNA regions, while the thinner lines are single-stranded. When two molecules come in vicinity of each other (Figure 4.7.1 (b) top), they combine to form the nanostructure shown (Figure 4.7.1 (b) bottom). The ssDNA we display in the above snapshots are $20 - 30\ bp$.

4.8 Discussion and Future work

We presented a comprehensive framework for building a software tool for simulating a DNA based molecular system, and not the actual software tool itself.

We believe that the methods presented here make a good framework for designing the simulator for DNA based molecular systems. We have described how to capture geometric constraints of the molecules with the polymer theory and MC simulation. The preliminary results in the chapter support the feasibility of the approach. We also described the approximations and limitations in this framework and the ways of improving them.

It is important to note that, as a framework, the physical simulation component and event simulation component can be decoupled as we improve each component individually.

Various improvements to simulation model can be made to improve the accuracy of the physical simulation:

- To reflect topological constraints by modeling more complicated DNA nanostructures such as pseudo-knots [78, 27].
- To provide more biophysical sound behavior of DNA strands by considering stacking energy and electrostatic energy .
- To achieve the molecular details by replacing the MC simulation with a BD simulation once computational resources are available.
- To validate its correctness against polymer theory and experimental data (for example, the average radius of gyration and the diffusion constant) and to update the physical simulation component to result in more realistic simulation.

A further extension to our framework would be to consider more complicated interactions, i.e. the enzyme restriction event and the hairpin formation. Another extension is to incorporate sequence design capabilities. We would like to design and optimize sequences based on the given nanostructure conformations. Furthermore, a conformation change of a nanodevice can be decomposed into units of local deformations to ease the sequence design.

Chapter 5

Autonomous Programmable DNA Nanorobotic Devices Using DNAzymes

A major challenge in nanoscience is the design of synthetic molecular devices that run *autonomously* (that is, without externally mediated changes per work-cycle) and are *programmable* (that is, their behavior can be modified without complete redesign of the device). DNA-based synthetic molecular devices have the advantage of being relatively simple to design and engineer, due to the predictable secondary structure of DNA nanostructures and the well-established biochemistry used to manipulate DNA nanostructures. However, ideally we would like to minimize the use of protein enzymes in the design of a DNA-based synthetic molecular device. We present the design of a class of DNA-based molecular devices using DNAzyme. These DNAzyme based devices are autonomous, programmable, and further require no protein enzymes. The basic principle involved is inspired by a simple but ingenious molecular device due to Mao et al [175] that used DNAzyme to traverse on a DNA nanostructure, but was not programmable in the sense defined above (it did not execute computations).

Our DNAzyme based designs include (1) a finite state automata device, *DNAzyme FSA* that executes finite state transitions using DNAzymes, (2) extensions to it including probabilistic automata and non-deterministic automata, and (3) its application as a *DNAzyme router* for programmable routing of nanostructures on a 2D DNA addressable lattice. Furthermore, we give a medical-related application, *DNAzyme doctor* that provide transduction of nucleic acid expression: it can be programmed to respond to the under-expression or over-expression of various strands of RNA, with a response by release of an RNA (The behavior of our nucleic acid transduction devices is similar to those of the prior paper

of Shapiro[21], but ours have the advantage that they operate without use of any protein enzymes.) In addition, we describe some background theory and mathematical models essential to the operation of our devices, including stochastic models for simulation of strand displacement, and the operation of DNAzyme.

5.1 Introduction

5.1.1 Prior Autonomous Molecular Computing Devices

In the last few years the idea of constructing complex devices at the molecular scale using synthetic materials such as DNA has gone from theoretical conception to experimental reality.

DNA Tiling Assemblies.

One theoretical concept that had considerable impact on experimental demonstrations was that of Wang Tiling; this is an abstract model that allows for a finite set of 2D rectangles with labeled sides to assemble 2D lattices by appending together tiles at their matching sides. Winfree first proposed the use of DNA nanostructures to achieve Wang Tiling computations; the DNA nanostructures known as *DNA tiles* that self-assemble into 2D lattices as determined by the tiles pads (ssDNA on the sides of the tiles that can hybridize to other tile's pads). The last decade has seen major successes in experimental demonstrations of the use of such DNA tiling assemblies to construct patterned lattices and tiling computations. DNA tiling assemblies have been used effectively in construction of periodic two-dimensional lattices, such as those made from double-crossover (DX) DNA tiles [195], rhombus tiles [109], triple-crossover (TX) tiles [94], and "4x4" tiles [202], as well as triangle lattices [104] and hexagonal lattices [41]. They have also been used for the construction of patterned lattices [201] by designing the DNA tile pads to program computations. The use of DNA tiling assembly has two major advantages over most other

methods for molecular computation, since it: (i) operates entirely autonomously, without outside mediated changes, and (ii) does not require the use of protein enzymes.

DNA tiling assemblies do have limitations: in particular, in general as currently conceived, they do not allow for the molecular devices (the tiles in their case) to transition between multiple states (except of course for their free or assembled states). In contrast, many complex molecular mechanisms found in the cell can transition into multiple states, allowing far more flexibility of application.

Autonomous Molecular Computing Devices that Execute Multiple State Transitions

There are only two other known methods for DNA computation that operate autonomously. Both use ingenious constructions, but require the use of enzymes.

(i) The *whiplash PCR machines* of [114, 121, 143, 191]. These however, can only execute a small number of steps before they require changes in the environment to execute further steps. Also, they require the use of polymerase enzyme.

(ii) The autonomous DNA machines of Shapiro[22, 20, 21], which execute finite transitions using restriction enzymes. The autonomous DNA machine [21] demonstrated molecular sensing and finite state response capabilities for that could be used for medical applications (though the demonstrations were made in test tubes only, rather than in natural biological environments as would be required for their medical applications). Their paper was important motivational factor in the work described here.

5.1.2 Our Main Contribution

This chapter provides the first known design for a DNA-RNA based devices that (a) operates autonomously, (b) do not require the use of protein enzymes, and (c) allow for the execution of multiple state transitions. Our designs make use of certain prior DNA nanomechanical devices, which will be discussed below.

5.1.3 DNA Nanomechanical Devices

Prior Nonautonoumous Nanomechanical DNA Devices

A variety of DNA nanomechanical devices have been constructed that exhibit motions such as open/close [162, 163, 210], extension/contraction [10, 58, 102], and rotation [110, 176, 203]. The motion of these devices is mediated by external environmental changes such as the addition and removal of DNA fuel strands [10, 58, 102, 162, 163, 176, 203, 210] or the change of ionic strength of the solution [110]. For example, non-autonomous progressive walking devices, mediated by the addition and removal of DNA strands, were constructed both by Seeman [160] and Pierce [161]. Although in many cases ingeniously designed, these devices need external (human or automation-based) intervention for each step of their motions. These synthetic DNA devices are in sharp contrast with cellular protein motors and machines on macroscale that operate autonomously, without requiring any interference.

Prior Autonoumous DNA Nanomechanical Devices

Recent times have seen significant progress in construction of DNA nanomechanical devices that execute autonomous, progressive motions. Reif [135] gave two designs for autonomous DNA nanomechanical devices that traverse bidirectionally along a DNA nanostructure. Turberfield et al proposed using DNA hybridization energy to fuel autonomous free-running DNA machines [178]. Peng et al [208] was the first to experimentally demonstrate an autonomous DNA walker, which is an autonomous DNA device in which a DNA fragment translocates unidirectionally along a DNA nanostructure. It used DNA ligase and restriction enzymes.

Recently Mao demonstrated two autonomous DNA nanomechanical devices driven by DNA enzymes (non-protein), namely (a) a tweezer [46, 45] which is a DNA nanostructure that open and closes autonomously and (b) a DNA crawler [175] using DNA enzyme,

which traverses across a DNA nanostructure.

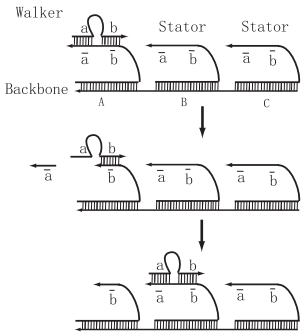


Figure 5.1: Overview of Mao's crawler [175] constructed using DNA enzyme

Their crawler device contains a DNA enzyme (DNAzyme) that constantly extracts chemical energy from its substrate molecules (RNA) and uses this energy to fuel the motion of the DNA device. This DNAzyme-based crawler integrates DNAzyme activity and strand-displacement reaction. They use 10-23 DNAzyme, which is a DNA molecule that can cleave RNA with sequence specificity. The 10-23 DNAzyme contains a catalytic core and two recognition arms that can bind to a RNA substrate. When the RNA substrate is cleaved, the short fragment dissociate from the DNAzyme and that provides a toehold for another RNA substrate to pair with short recognition arm of the DNAzyme. The crawler device traverses on a series of RNA stators implanted on a nanostructure as shown in Figure 5.1.

Their crawler is the primary inspiration to our designs. While an ingenious device, there are a number of limitations of Mao's DNAzyme-based crawler: (1) it did not demonstrate the loading and unloading of nanoparticles (2) it only traverses along a one dimensional sequence of ssRNA strands (stators) dangling from a DNA nanostructure, and its route is not programmable (3) it does not execute finite state transitions beyond what are required to move (that is, it does not execute computations).

5.1.4 Overview of this Chapter and Results

The goal of this chapter is to address the above limitations, providing substantially enhanced functionalities to the prior DNAzyme-based crawler previously developed. All the devices described in this chapter are based on selective cleaving activity of DNAzyme and strand displacement processes. For the purpose of modeling these devices, it becomes imperative to model the strand displacement and cleaving activity of DNAzymes first. In the Section 5.2, we describe the kinetic models for these processes. We present the design of *DNAzyme FSA*: a finite state machine based on the activity of DNAzyme and strand displacements in Section 5.3. DNAzyme FSA can be easily extended to non-deterministic finite state automata and probabilistic automata as described in Section 5.3.7 and 5.3.8. In Section 5.4 we present a medical related application of DNAzyme FSA referred to as *DNAzyme doctor*. DNAzyme doctor is a molecular computer for logical control of RNA expression using DNAzyme. Another application of DNAzyme FSA, *DNAzyme router*: a DNAzyme based system for programmable routing of the walker on a 2D lattice is described in Section 5.5.

5.2 Strand Displacement and DNAzyme

The devices described in this chapter are based on selective cleaving activity of DNAzyme and strand displacement processes. For the purpose of modeling these devices, it becomes imperative to model the strand displacement and cleaving activity of DNAzymes first. In the next subsections we describe the kinetic models for these processes.

5.2.1 Strand Displacement

In a strand displacement process two strands compete against each other to hybridize with a third strand. Figure 5.2.1 shows a strand displacement process where strand B and C are

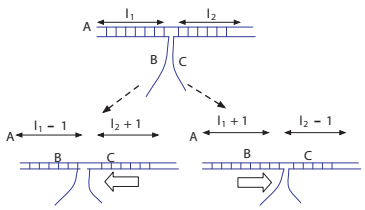


Figure 5.2: Strand displacement: molecule B and C compete against each other to hybridize with molecule A

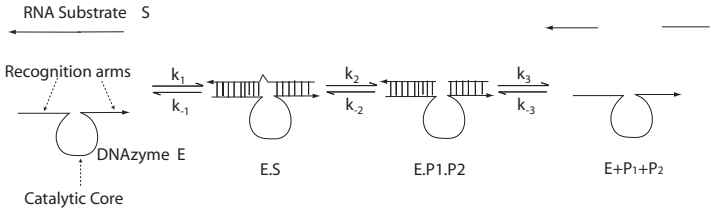


Figure 5.3: Mechanism of the cleaving of RNA substrate by DNAzyme

competing against each other to hybridize with strand A. This ultimately results in removal of one of the competing strands, and hence the term strand-displacement.

Strand displacement can be modeled as a random walk of the junction where the two strands are competing against each other. For every step, the direction of migration of this junction is chosen probabilistically independent of its previous movements. It has been shown that the strand displacement is a biased random walk in case of base-pair mismatches in these strands [25]. In other words, migration probability towards the direction with base-pair mismatches is substantially decreased.

Let us denote the nanostructure shown on top in Figure 5.2.1 as molecule ABC. Let G°_{ABC} be its free energy. Denote G°_{rABC} and G°_{lABC} as the free energy of ABC after 1 base pair migration towards right, and left, respectively. Let $\Delta G^\circ_r = G^\circ_{rABC} - G^\circ_{ABC}$ and $\Delta G^\circ_l = G^\circ_{lABC} - G^\circ_{ABC}$. Let p_r be the probability of the right-directional migration and p_l be the probability of the left-directional migration. It has been shown in [25] that $p_r \propto \exp(-\Delta G^\circ_r/RT)$, similarly $p_l \propto \exp(-\Delta G^\circ_l/RT)$, where the change of free energies can be computed by the NN model [83]. Thompson et al[174] calculated the average time

taken per base-pair migration (time per step) to be of the order of 100 μ sec. Strand displacement processes can be modeled as discrete time Markov chain processes using the above mentioned parameters.

5.2.2 DNAzymes

DNAzyme (also known as deoxyribozymes, DNA enzymes, and catalytic DNA) is a DNA molecule with a catalytic action. One of the widely used DNAzyme is 10-23 DNAzyme. It can cleave RNA with sequence specificity. The 10-23 DNAzyme contains a catalytic core and two recognition arms that can bind to a RNA substrate as shown in Figure 5.2.1. The recognition domains provide both the sequence information necessary to specify RNA substrate and the binding energy needed to hold the substrate within the active site of enzyme. When the RNA substrate is cleaved, the short fragments dissociate from the DNAzyme. Another well studied DNAzyme is 8-17 deoxyribozyme. It also contains a catalytic core and two recognition arms. It is comparatively less flexible as compared to 10-23 DNAzyme in terms of target choice: 10-23 can cut an RNA phosphodiester bond located at any purine-pyrimidine site, while 8-17 requires an AG or GG site[39, 79].

Kinetics and thermodynamic characterization for cleaving activity of 8-17 DNAzyme and 10-23 DNAzyme are described in details in [28] and [152], respectively. RNA-cleaving activity of DNAzyme can be usually described into three reversible steps as shown in the Figure 5.2.1. First step is the hybridization of the enzyme with the substrate. The second step is the cleaving of the substrate by the enzyme, which always requires metal ions as cofactor. This is usually the rate determining step in the reaction. Third step is the release of the cleaved product. k_1, k_2, and k_3 are the respective forward rate constants, and k_{-1}, k_{-2}, and k_{-3} are the respective reverse rate constants for the above mentioned three reversible steps. Substrate cleavage rate, $k_2 >> k_{-2}$ (ligation rate) suggests that enzyme has a strong preference for substrate cleavage over ligation. Enzyme substance association rate,

$k_1 >> k_{-1}$ (dissociation rate) which is responsible for high enzyme-substrate association and hence high catalytic efficiency. Rate of product release step, $k_3 >> k_2$, which shows that substrate cleavage (rate constant k_2) is the rate determining step. It has been shown that DNAzymes show a high degree of sequence specificity. Catalytic rate increases log-linearly with increasing pH and linearly with various metal divalent cations. Under higher pH and modified divalent cation conditions, the 10-23 DNAzymes can cleave with k_{cat} of ~ 10 min^{-1} and a catalytic efficiency, k_{cat}/K_M, of 10^9 M^{-1}min$^{-}1$[152].

5.3 DNAzyme FSA: DNAzyme Based Finite State Automata

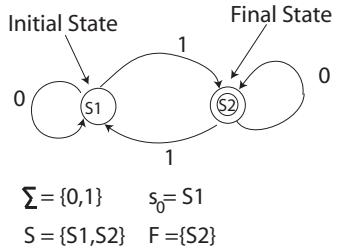


Figure 5.4: A finite state automata

A *finite state automata* can be described as a 5-tuple $(\Sigma, S, s_0, \delta, F)$, where Σ is a finite non-empty set of symbols called input alphabet, S is a finite non-empty set of states, $s_0 \in S$ is an initial state, δ is the state transition function ($\delta : S \times \Sigma \rightarrow S$), and $F \subset S$ is the set of final states.

Figure 5.4 illustrates an example of a finite state automata that accepts a binary string containing odd number of 1s.

In this section, we describe a DNAzyme based finite state automata, referred to as DNAzyme FSA. At any time an RNA sequence encoding an input symbol is examined by the DNAzyme FSA, then an appropriate state transition takes place, and then the RNA sequence encoding the next input symbol is examined. This process continues till all the

input symbols are scanned and the output of the DNAzyme FSA is its state at the end of process.

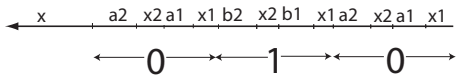


Figure 5.5: Encoding of 0 and 1 in DNAzyme FSA

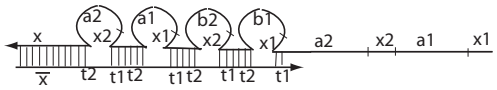


Figure 5.6: Protector strand partially hybridizes with the input strand to form bulge loops. The sticky end formed at the end of the input strand outside of the bulge loops represents the active input symbol. This scheme protects the input symbols other than the currently active symbol from becoming active

5.3.1 Encoding the Input Symbols

First of all, we describe the way the input is encoded for the DNAzyme FSA. Input symbols 0 and 1 are encoded as the RNA sequences $x_1 \cdot a_1 \cdot x_2 \cdot a_2$ and $x_1 \cdot b_1 \cdot x_2 \cdot b_2$, respectively, where a_1, a_2, b_1, b_2, x_1, and x_2 are RNA sequences, and $\cdot$ represents concatenation. Figure 5.3 illustrates this encoding of the input symbols. It should be noted that 0 and 1 share common subsequences x_1 and x_2. Also, there is a special subsequence x at the end of the input subsequence. This is central to the working of the DNAzyme FSA as will be explained later.

5.3.2 Active Input Symbol

While encoding the input for DNAzyme FSA, it is essential to have a mechanism to detect the current input symbol that is being scanned by DNAzyme FSA. We will refer to this symbol as *active input symbol*. In order to implement this feature in DNAzyme FSA only a small segment of the RNA strand encoding the input symbols is kept active. Most part of

it is kept protected by hybridization with a partially complementary sequence, referred to as *protecting sequence*. It has not been shown in the Figure 5.3 but the protecting sequence should not be one continuous strand. Instead it should contain nicks at various positions. This is necessary for the working of device and will be explained later. The active input symbol is represented by the sticky end of the RNA sequence encoding the input. We refer to this nanostructure as *input nanostructure*. Figure 5.3 illustrates the idea. The input nanostructure encodes the input 010. The active input symbol is rightmost 0 (in 010), and it is encoded by the sticky end of the input nanostructure, and hence is active. However, the leftmost 0 and the 1 are encoded in the protected portion of the input nanostructure. They have been protected by hybridization with a protecting sequence. Since the protecting sequence is partially complementary to the RNA sequence encoding the input symbols, it results in the formation of bulge loops. In the Figure 5.3 a_2, a_1, b_2, and b_1 contain a subsequence complementary to t_2, while x_2 and x_1 contain subsequence complementary to t_1. Since the RNA sequence encoding input is partially complementary to the protecting sequence $t_2.t_1.t_2.t_1...$ it forms the bulge loop structure as shown in the Figure 5.3. Each input symbol is hence represented by two bulge loops. It should be noted that the special sequence x at the end of the input sequence and $\bar{x}$ at the end of protecting sequence ensure that only the desired alignment of protecting sequence with input sequence is favored. As a result, only the desired input nanostructure as shown in Figure 5.3 is formed.

5.3.3 States and Transitions

After the description of the input, next we describe the design of states and transitions in finite state machine. In DNAzyme FSA, a network of DNAzymes is embedded on a two-dimensional plane, and the input nanostructure is routed over it. The state of the DNAzyme FSA at any time is indicated by the DNAzyme that holds the input nanostructure at that time. During each state transition of DNAzyme FSA, the segment of input nanostructure

encoding the active input symbol is cleaved, the next bulge loop opens up exposing the segment encoding next input symbol, thereby making it new active input symbol, and the input nanostructure jumps to another DNAzyme that indicates the new state of DNAzyme FSA. In subsequent paragraphs, we will explain in details the complete process of state transition in DNAzyme FSA.

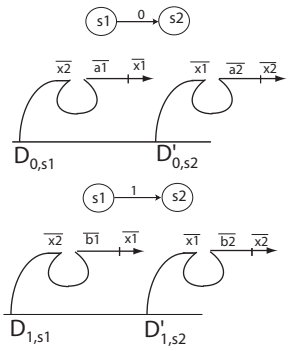


Figure 5.7: Figure illustrates the implementation of a state transition through DNAzymes

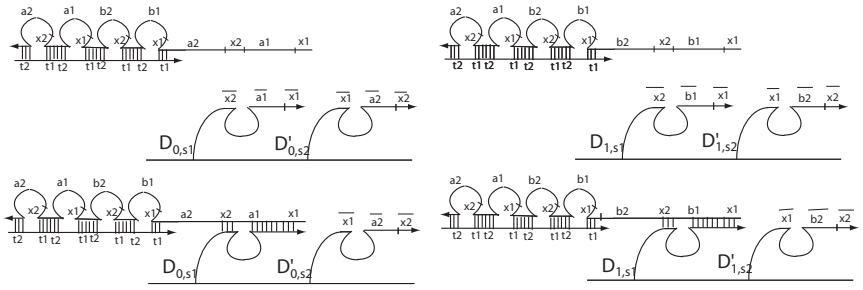


Figure 5.8: D_{0,s_1} in the transition machinery for state transition at 0 combines with input nanostructure when active input symbol encoded by the sticky end is 0. When the active input symbol encoded by the sticky end is 1, D_{1,s_1} in the transition machinery for state transition at 1 combines with the input nanostructure

As shown in Figure 5.7 (a), a state transition from one state to another is implemented as two evenly spaced DNAzymes, referred to as *transition machinery* for that state transition. Each of these DNAzymes is tethered to another DNA nanostructure, which forms part of the backbone of the DNAzyme FSA. DNAzyme D_{0,s_1} and D'_{0,s_2} form the transition machinery for state transition from state s_1 to state s_2 for input 0. Similarly, DNAzyme

D_{1,s_1} and D'_{1,s_2} form the transition machinery for state transition from state s_1 to state s_2 for input 1. It should be noted that in our nomenclature the first subscript of the DNAzyme specifies the active input symbol and the second subscript specifies the states for a transition machinery.

The foremost thing to ensure in DNAzyme FSA is that if the active input symbol is 0, then the state transition for input 0 should be taken. Similarly, if the active input symbol is 1, then the state transition for input 1 should be taken.

In the transition machinery for state transition for input 0, the DNAzymes D_{0,s_1} and D'_{0,s_2} contain DNA subsequences $\overline{x_2} \cdot \overline{a_1} \cdot \overline{x_1}$ and $\overline{x_1} \cdot \overline{a_2} \cdot \overline{x_2}$ respectively, at their free ends. The DNA subsequences of D_{0,s_1} is partially complementary to the RNA sequence that encode the symbol 0 ($x_1 \cdot a_1 \cdot x_2 \cdot a_2$). This ensures that only when the sticky end of input nanostructure is $x_1 \cdot a_1 \cdot x_2 \cdot a_2$, it can hybridize with the DNAzyme D_{0,s_1}. Thus a state transition for 0 is not taken in DNAzyme FSA, unless the active input symbol is 0.

Similarly, in the transition machinery for state transition for input 1, the DNAzymes D_{1,s_1} and D'_{1,s_2} contain DNA subsequences $\overline{x_2} \cdot \overline{b_1} \cdot \overline{x_1}$ and $\overline{x_1} \cdot \overline{b_2} \cdot \overline{x_2}$ respectively, at their free ends. These subsequences are partially complementary to the RNA sequence that encode the symbol 1 ($x_1 \cdot b_1 \cdot x_2 \cdot b_2$). As explained earlier, this ensures that a state transition for 1 is not taken in the DNAzyme FSA, unless the active input symbol is 1. Figure 5.8 further illustrates the idea.

5.3.4 Description of State Transition

In this section, we will describe the movement of the input nanostructure over the DNAzymes in a transition machinery to carry out the state transition in DNAzyme FSA. Figure 5.9 shows a transition machinery for input 0. Initially, the input nanostructure is hybridized with the DNAzyme D_{0,s_1}. The sticky end of the input nanostructure represents the active input symbol 0, and therefore, the transition at input 0 is to be performed. First, the

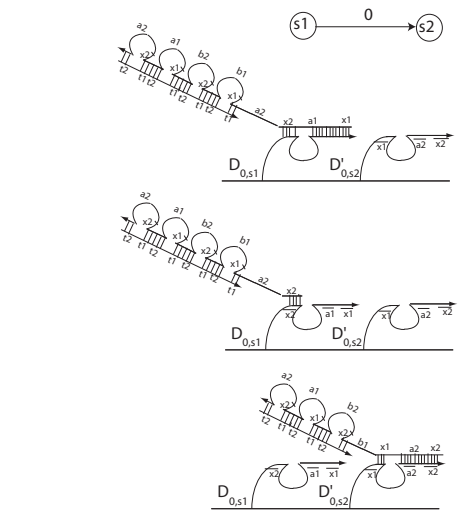


Figure 5.9: First half of a state transition by DNAzyme FSA from s_1 to s_2 at input 0 is illustrated. Sequence encoding active input symbol 0 gets cleaved by DNAzyme D_{0,s_1}, input nanostructure moves to next DNAzyme D'_{0,s_2} by strand displacement, and the next bulge loop in the input nanostructure opens up in the process

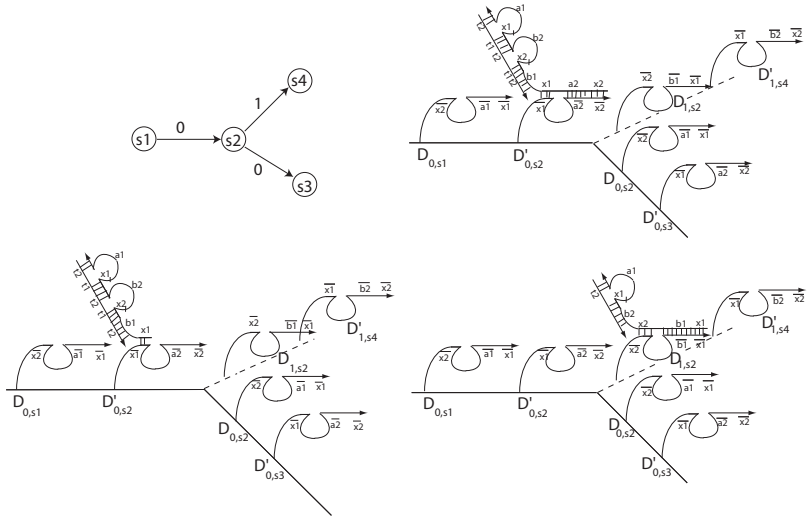


Figure 5.10: Second half of a state transition by DNAzyme FSA from s_1 to s_2 at input 0 is shown. The mechanism is similar to the first half. However, in this part the next input symbol and next state transition of DNAzyme FSA is determined, and the input nanostructure lands up on the appropriate transition machinery for the next state transition to begin correctly

DNAzyme D_{0,s_1} cleaves the input nanostructure as shown in Figure 5.9. Now the sticky end of input nanostructure has only x_2 as complementary subsequence to the subsequence

$\overline{x_2} \cdot \overline{a_1} \cdot \overline{x_1}$ at the free end of DNAzyme D_{0,s_1}. However, the longer subsequence $x_2 \cdot a_2$ in its sticky end is complementary with the subsequence $\overline{a_2} \cdot \overline{x_2}$ of DNAzyme D'_{0,s_2}. Therefore, a strand displacement process takes place with the free ends of DNAzymes D_{0,s_1} and D'_{0,s_2} competing against each other to hybridize with sticky end ($x_2 \cdot a_2$) of the input nanostructure. Since D'_{0,s_2} provides a longer complementary subsequence, ultimately D_{0,s_1} is displaced and the input nanostructure is now hybridized with D'_{0,s_2} as shown in Figure 5.9. It should be noted that the next bulge loop gets opened in this process. An input symbol is encoded across two bulge loops in the input nanostructure. As the first half of the sticky end ($x_1 \cdot a_1$) encoding the half of the active input symbol 0 got cleaved, the current sticky end is $x_2 \cdot a_2 \cdot x_1 \cdot b_1$, that contains half of the sequence encoding symbol 0 and half of the sequence encoding the symbol 1. This completes the first half of the state transition by DNAzyme FSA.

The second half of the transition in DNAzyme FSA takes place in exactly similar manner. Half of the sticky end ($x_2 \cdot a_2$) of the input nanostructure that encodes the remaining half of the active input symbol 0 gets cleaved, thus leaving only x_1 as complementary to free end of DNAzyme D'_{0,s_2} ($\overline{x_1} \cdot \overline{a_2} \cdot \overline{x_2}$). At this point the sticky end of the input nanostructure is $x_1 \cdot b_1$ which is half of the sequence that encodes the input symbol 1. It indicates that the next active input symbol is 1 and therefore, the next state transition should be from state s_2 at input 1. This is ensured by the DNAzyme FSA in the following way. Since the sticky end of the input nanostructure is ($x_1 \cdot b_1$), the DNAzyme D_{1,s_2} that has the sequence $\overline{x_2} \cdot \overline{b_1} \cdot \overline{x_1}$ at its free end gets involved in strand displacement with D'_{0,s_2} to hybridize with the sticky end ($x_1 \cdot b_1$) of input nanostructure. Because of the longer complementary sequence D_{1,s_2} ultimately displaces D'_{0,s_2} and hybridizes with the sticky end of nanostructure. This results in the opening of next bulge loop in input nanostructure as shown in Figure 5.10 .

It should be noted that D_{0,s_2} (with sequence $\overline{x_1} \cdot \overline{b_2} \cdot \overline{x_2}$ at its free end) does not have sequences complementary to the sticky end ($x_1 \cdot b_1$) of input nanostructure, so it can not

get involved in any strand displacement. Therefore, the input nanostructure is guaranteed to move to the DNAzyme D_{1,s_2}. After the opening of the next bulge loop, the new sticky end ($x_1 \cdot b_1 \cdot x_2 \cdot b_2$) of input nanostructure encodes the input symbol 1. Thus, the input nanostructure lands up in the appropriate transition machinery for the next state transition, and the next state transition at input 1 can begin correctly.

It can be argued in a similar manner that during the second half of the transition, if the next active input symbol was to be 0, the input structure would have moved from DNAzyme D'_{0,s_2} to D_{0,s_2} instead of moving to D_{1,s_2}. We omit the explanation here for the sake of brevity.

Figure 5.10 illustrates the second half of the state transition of DNAzyme FSA.

It should be noted that the strand displacement of the protector strand also takes place during the process. But since it contains nicks, its fragments just wash away in the solution when they get completely displaced.

5.3.5 Complete State Machine

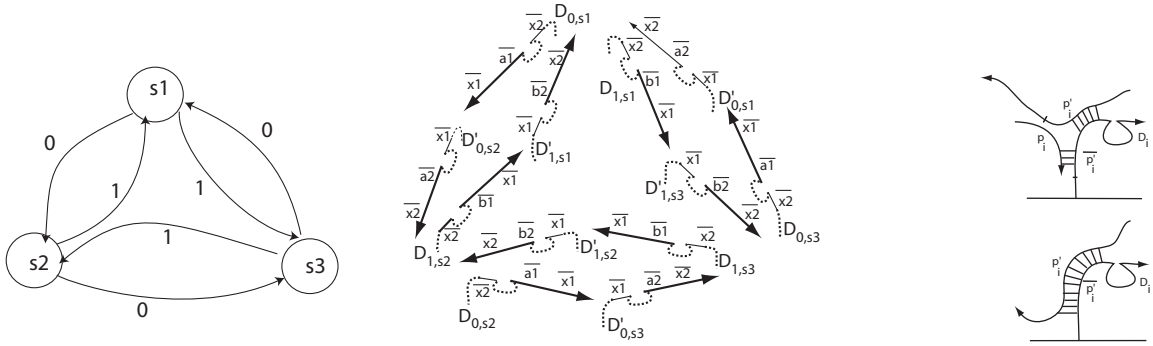


Figure 5.11: (a) The DNAzyme implementation of the finite state machine shown on left. (b) Reporting sequence displaces the probe strand from the stem of the DNAzyme that indicates the output state of DNAzyme FSA. Thus, the output can be detected using fluorescent in-situ hybridization technique

The components described above can be integrated to implement the complete finite

state automata. Any state transition in the DNAzyme FSA can be implemented by two DNAzymes as described earlier. These DNAzymes are embedded on a nanostructure that forms the backbone of the DNAzyme FSA. The addressable nanostructures formed by DNA origami [144] or fully-addressable DNA tile lattices [125] might provide useful nanostructures for this backbone. Hence, the state machine can be laid out on this nanostructure by implanting a network of DNAzymes on it. The input nanostructure traverses over them in a programmable way and keeps getting cleaved in the process.

Figure 5.11 (a) shows an implementation of a DNAzyme FSA (at the right) for the finite state automata (at the left). It should be noted that the DNAzymes shown in the Figure 5.11 (a) are actually implanted on a backbone nanostructure. The dashed lines represent the sides of these DNAzymes that are embedded in the backbone nanostructure.

The output of the DNAzyme FSA is detected using insitu hybridization techniques. The details of the protocol are described in Section 5.3.6.

5.3.6 Detecting the Output State

In this section we describe the technique to detect the output of the DNAzyme FSA after completion of the computation on a given input. The state of DNAzyme FSA at the end of the computation is the output state. A special sequence is incorporated inside the last bulge loop in the input nanostructure. We call it as *reporting sequence*. In the DNAzyme FSA described above, as the state transitions take place, the input nanostructure gets cleaved and the bulge loops open up one by one as explained earlier. When the input gets cleaved upto the reporting sequence, the computation is completed. At this time the reporting sequence becomes available, and its position on the DNAzyme FSA indicates the output state. The role of reporting sequence in the detection of final state is described below.

Fluorescent in-situ hybridization (FISH) [30, 153, 164] is a cytogenetic technique which can be used to detect and localize the presence or absence of specific DNA se-

quences on longer DNA strands. It uses fluorescent probes which bind only to those parts of the DNA strands with which they show a high degree of sequence complementarity. All unhybridized or partially hybridized probes disappear, and only the probes that hybridized to the target are visible in fluorescence.

In our DNAzyme FSA, the stems of the DNAzyme stators contains a unique DNA subsequence. Let us assume that DNAzyme D_i contains the sequence $\overline{p_i'}$. The reporting sequence of the input nanostructure is designed in such a way so that it has segments complementary to each of these $\overline{p_i'}$ subsequences. Hence, reporting sequence is essentially a concatenation of p_i's. At the same time we have different probes each corresponding to a different DNAzyme stator. Each of them is labeled with a different fluorescent dye. It should be noted that probe p_i that is attached to DNAzyme D_i is a subsequence of p_i' as shown in Figure 5.11 (b).

As mentioned earlier that the reporting sequence becomes available at the end of the computation. In case D_i is the DNAzyme with which the reporting sequence is hybridized at one end, D_i determines the the output state. Since p_i' of the reporting sequence is a better complement to $\overline{p_i'}$ of DNAzyme D_i as compared to the probe p_i. Therefore, the reporting sequence displaces the probe p_i that contains the fluorescent dye from DNAzyme D_i, and p_i disappears in the solution. Figure 5.11 (b) illustrates this process.

Hence, all other probes except the one that hybridized to the DNAzyme determining the output state are visible in fluorescence. This protocol can be used to detect the output state of the automata.

5.3.7 Non-deterministic DNAzyme FSA

A *nondeterministic finite state automata* is a 5-tuple $(\Sigma, S, s_0, \delta, F)$, where Σ is a finite set of input symbols, S is a finite set of states, δ is a state transition function $(\delta : S \times (\Sigma \bigcup \{\epsilon\}) \rightarrow P(S)$ where $P(S)$ is the power set of S), ϵ is the empty string, $s_0 \subset S$ is

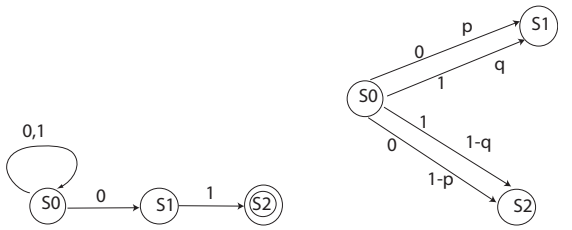


Figure 5.12: (a) A non-deterministic finite automata that accepts $(0+1)^*01$ (b) Schematic of a probabilistic automata. The transition from state S_0 on input 0 takes place to state S_1 with probability p and to state S_2 with probability $1-p$. Similarly, the transition from state S_0 on input 1 takes place to state S_1 with probability q and to state S_2 with probability $1-q$. p and q are real numbers between 0 and 1

a set of initial states, and $F \subset S$ is a set of final states. Figure 5.12 (a) shows one non-deterministic automata that accepts the language $(0+1)^*01$ (the set of binary strings that ends with "01").

The idea extends to the non-deterministic automata directly. Different DNAzyme-FSA described above will work in parallel inside a test-tube. Therefore, the above described scheme will work for non-deterministic automata as well. In case there are more than one transitions possible for one input from one state, each of them will be taken in one DNAzyme-FSA or the other inside the solution, and thus exhibiting non-deterministic nature of the automata. Regarding the output, if the output state in any of the DNAzyme-FSA in solution is an accepting state (or final state), it implies the acceptance of the input by the overall non-deterministic finite state automata.

5.3.8 Probabilistic DNAzyme FSA

A *probabilistic finite state automata* is a finite state automata in which the state transitions are probabilistic in nature. It can be described as a 5-tuple $(\Sigma, S, s_0, \delta, F)$, where Σ is a finite set of input symbols, S is a finite set of states, δ is a state transition function ($\delta : S \times \Sigma \times S \rightarrow [0,1]$), $s_0 \subset S$ is a set of initial states, and $F \subset S$ is a set of final states. Figure 5.12 (b) shows a probabilistic automata.

In case the sequences of all the DNAzymes are identical, then the DNAzyme-FSA described above becomes a probabilistic automata having equal probabilities of transitions from any state to any other state. However, to construct an arbitrary probabilistic finite state automata, the probabilistic transitions can be implemented by using partially complementary sequences in the designs. The sequences of the DNAzymes for transition are chosen in a way so that the ratios of probability of hybridization are in accordance with the transition probabilities.

5.4 DNAzyme Doctor: A Molecular Computer for Logical Control of RNA Expression using DNAzyme

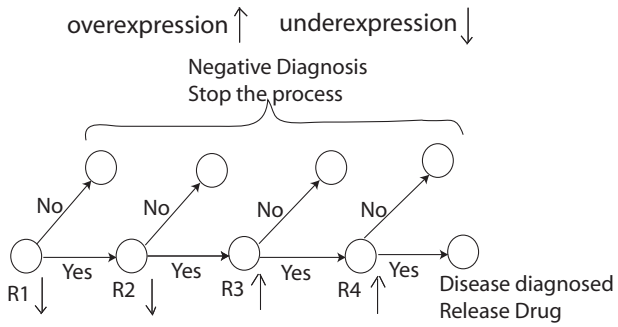


Figure 5.13: A state diagram for DNAzyme doctor that controls the release of a drug RNA on the basis of the RNA expression tests for the a disease

The finite state automaton described in Section 5.3 can be used in various computational and routing applications. In this section we describe DNAzyme doctor, an application related to medical field. It is an autonomous molecular computer for control of RNA expression based on the overexpression and underexpression of other RNAs. Earlier Shapiro[21] had constructed a molecular computer using protein enzymes for logical control of RNA expression. Their molecular computer analyses the levels of messenger RNA species and in response produces a molecule capable of affecting levels of gene ex-

pression. DNAzyme doctor performs the same function, while completely eliminating the use of protein enzymes in the design. For the ease of illustration let us consider a similar example as given in [21]. Suppose a disease is diagnosed positive if RNAs R_1 is underexpressed, R_2 is underexpressed, R_3 is overexpressed, and R_4 is overexpressed. Thus, the detection of the disease can be done by computing logical AND of the above mentioned four RNA expression tests. In case it is established that the disease exists, a curing drug should be released. While in any other case, the drug should not be released. Figure 5.3.8 illustrates the aforementioned logic in the form of a state diagram. The sequences y_1, y_2,

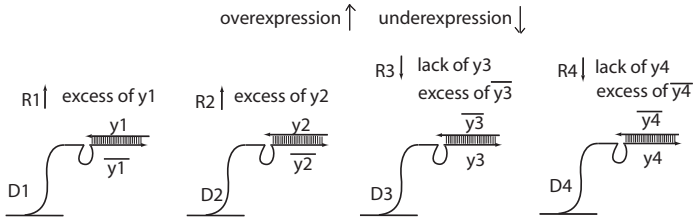


Figure 5.14: The figure shows the consequences of overexpression and underexpression of different RNAs on the concentrations of the respective characteristic sequences. The overexpression of R_1 and R_2 results in excess of y_1 and y_2 respectively, and they block the path of input nanostructure by hybridizing with D_1 and D_2. Similarly underexpression of R_3 and R_4 results in excess of $\overline{y_3}$ and $\overline{y_4}$ respectively, to block the path of input nanostructure

y_3 and y_4 are characteristic sequences of RNAs R_1, R_2, R_3, and R_4 respectively. The levels of RNA R_1, R_2, R_3, and R_4 are

The concentrations of the characteristic sequences y_1, y_2, y_3, y_4 as well as their complements $\overline{y_1}$, $\overline{y_2}$, $\overline{y_3}$, $\overline{y_4}$ are regulated.

If R_1 is overexpressed then y_1 is in excess, and if R_2 is overexpressed then y_2 is in excess. However, if R_3 is underexpressed, then lack of y_3 and if R_4 is underexpressed, then lack of y_4. But a threshold concentration of $\overline{y_1}$, $\overline{y_2}$, $\overline{y_3}$, $\overline{y_4}$ is thrown into the solution, therefore lack of y_3 causes excess of $\overline{y_3}$, and lack of y_4 causes excess of $\overline{y_4}$.

Since the DNAzyme doctor only needs to perform a logical AND, it can be implemented in a simple way. We make the input nanostructure walk over four DNAzyme

stators implanted on a nanostructure in a straight path (as shown in Figure 5.15). Each DNAzyme stator represents one of the RNA expression test. In case the test is positive, the input nanostructure moves to next DNAzyme stator, otherwise it gets stuck and ultimately floats away in the solution. Therefore, the successful traversal of input nanostructure over all these DNAzyme stators implies that all tests are positive, and hence positive diagnosis of the disease.

In case the first test is negative (ie. overexpression of R_1), then excessively floating y_1 can bind to $\overline{y_1}$ part of the DNAzyme D_1. Similarly if second, third, or fourth tests are negative (ie.. overexpression of R_2, underexpression of R_3 or underexpression of R_4), then excessively floating y_2, y_3, or y_4 can bind to $\overline{y_2}$, $\overline{y_3}$, $\overline{y_4}$ portions of DNAzyme D_2, D_3, or D_4, respectively. The principle idea is illustrated in Figure 5.14.

Figure 5.15 shows the details of the sequences used in the design. It should be noted that $\overline{a_i} \cdot \overline{x_i}$ is a subsequence of $\overline{y_i}$. The input nanostructure traverses over the DNAzymes step by step as shown in Figure 5.15. The underlying mechanisms of these steps has been explained in Section 5.3. As explained earlier, when the input nanostructure moves to next DNAzyme, some portion of the sticky end is cleaved, and the next bulge loop opens up to restore the length of the sticky end. As can be seen in Figure 5.15, after the DNAzyme D_3 cleaves the sticky end of input nanostructure, the input structure moves to DNAzyme D_4, and the last bulge loop in input nanostructure opens up. The last bulge loop in the input contains a *drug-release trigger*. After the cleaving action by DNAzyme D_4, the drug-release trigger part of input structure is loosely bound with D_4. The drug-release trigger is then released in the solution. The actual drug is kept protected in the solution, as shown in Figure 5.14. The drug-release trigger displaces the *lock* strand from the nanostructure that hides the drug as shown in Figure 5.14.

It should be noted that if any of the tests are negative then the traversal of input nanostructure over the path of DNAzymes is blocked. Hence, if the ith test fails, then the

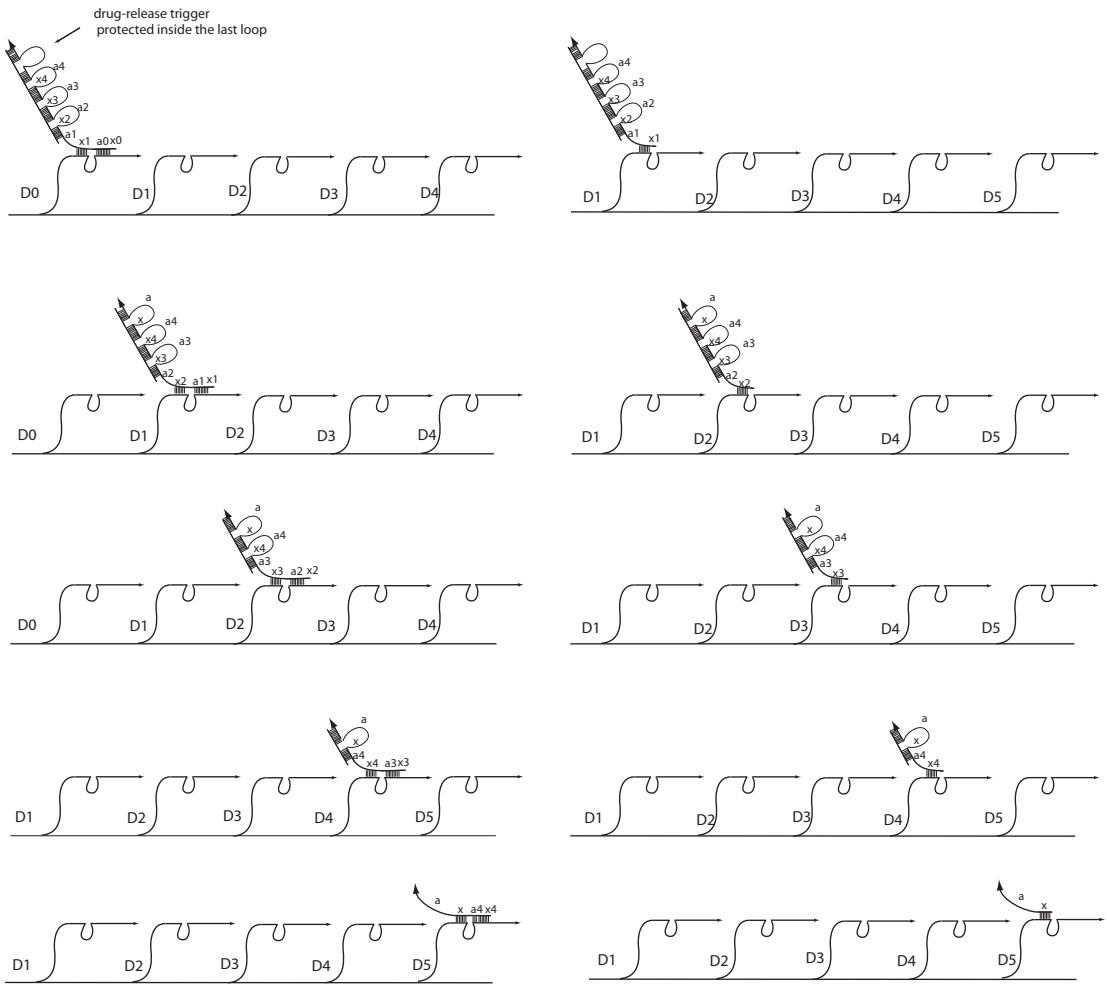


Figure 5.15: The input structure walks over the DNAzyme structures D_1, D_2, D_3, and D_4 as explained in Section 5.3. The drug to be released in case of positive diagnosis of the disease is protected within the last bulge loop of input structure

DNAzyme D_i is already hybridized with the DNA sequence y_is. It should be noted that $\overline{a_i} \cdot \overline{x_i}$ is a subsequence of $\overline{y_i}$. The DNAzyme D_i already hybridized with y_i would not participate in strand displacement with previous DNAzyme D_{i-1} to hybridize with sticky end of input nanostructure. Therefore, the input nanostructure can not traverse across this DNAzyme D_i and gets blocked at D_{i-1}.

After the cleaving of half of the sticky end of input nanostructure by DNAzyme D_{i-1}, its binding with D_{i-1} is not too strong either. So finally it detaches from the current DNAzyme and floats away in the solution. Hence the input structure is not cleaved upto the

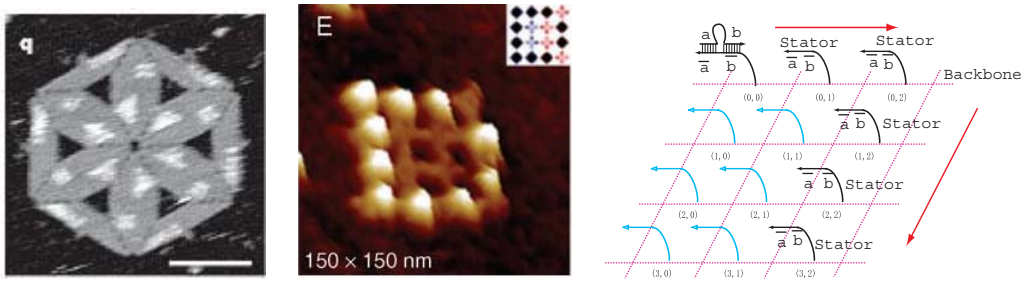
last bulge loop that contains the drug-release trigger, and therefore the drug-release trigger does not get released.

The ultimate goal in designing such a device will be to impart it the ability to perform inside a cell. It will require the device to be protected from other enzymes inside the cell. This protection can be imparted by embedding the device inside artificial lipozymes.

5.5 Application of DNAzyme for Routing

DNAzyme crawler can be routed on a two-dimensional DNA lattice in a naive manner as described in Section 5.5.1. The limitations posed by this simple routing scheme are overcome by DNAzyme router: a DNAzyme based system for programmable routing of the walker on a 2D lattice described in Section 5.5.2. DNAzyme router is an application based on the design of DNAzyme FSA described earlier in Section 5.3.

5.5.1 Routing DNAzyme Crawler in Two Dimensional Lattice



(a) (b) (c)

Figure 5.16: (a) A shape with pattern constructed using DNA origami by Rothemund [144](b) Letter D on a fully addressable 2D lattice constructed by Park et. al[125] (c) A predefined path on a fully addressable 2D DNA lattice for a DNAzyme crawler

Rothemund [144] developed a method for using scaffolded origami to create arbitrary nanoscale shapes (Figure 5.16 a)) which may be decorated with arbitrary nanoscale patterns. Also, fully addressable two-dimensional DNA tile lattices (Figure 5.16 b)) have been demonstrated [125]. Specific DNA strands can be mounted at desired locations on these

addressable nanostructures. Therefore, DNA stators can be embedded along any arbitrary path in a fully addressable 2D DNA lattice, as shown in Figure 5.16 c). DNAzyme crawler [175] can be made to travel along this predefined path of DNA stators, hence producing a motion in two-dimensions. However, in this scheme the path on which the DNAzyme crawler travels can be used only once.

The obvious advantage of this scheme is its simplicity. But the disadvantages are that the DNAzyme crawler can only travel on a predefined path and the path gets destroyed as the DNAzyme crawler moves along it. We present a more flexible and non-destructive scheme for two-dimensional routing, referred to as DNAzyme router in Section 5.5.2.

5.5.2 DNAzyme Router: DNAzyme based Programmable Routing in Two Dimensions

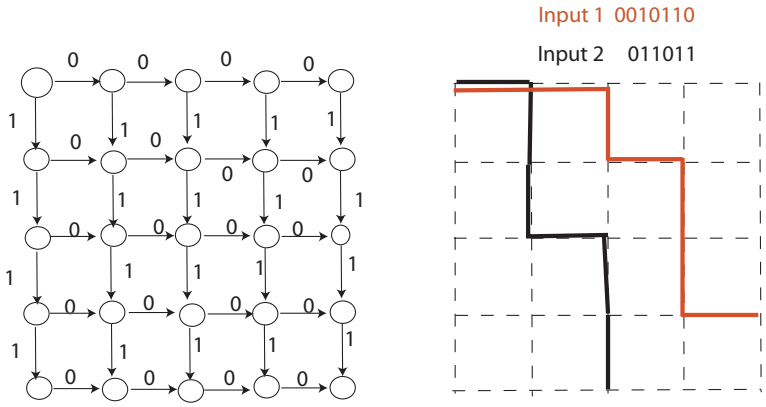


Figure 5.17: Illustration of programmable routing in two dimensions

For any arbitrary path along the network of DNAzymes in a given DNAzyme FSA, an input nanostructure can be designed to traverse along that path. This principle can be used for the design of a programmable routing system. The input nanostructure that moves over the DNAzyme FSA is referred to as *walker* and the complete system as DNAzyme router. The path of the walker is programmed through the state transitions of the automata and

the input symbols encoded in the walker. As an example, we can create a state machine on a rectangular grid (Figure 5.17), in which you move right if the input is 0, and towards bottom if the input is 1. Then by the state machine shown in Figure 5.17(a) and input shown in Figure 5.17(b) the walker can be made to travel along the path shown in Figure 5.17(b).

It should be noted that in a DNAzyme router the path does not get destroyed as a result of the motion of the walker. It is the input nanostructure (walker) that gets cleaved in the process, which is equivalent to exhaustion of fuel as a result of motion. Most remarkable feature of DNAzyme router is that we can have multiple walkers moving on the grid independently, each having its own programmed path.

5.6 Conclusion

We have described the construction of various devices based on the DNAzymes. In this chapter, we have focused more towards novel designs for the devices that can perform finite state transitions rather than the details of the laboratory implementation. However, it should be noted that among other implementational challenges, the construction of the desired bulge loops for input nanostructures needs further investigation. DNAzymes evolve through invitro selection procedures, and these processes can be designed to generate DNAzymes that cut distinct sequences. In the DNAzyme FSA, the number of DNAzymes required is proportional to the number of transitions in the automata. For binary-coded inputs the number of transitions is proportional to number of states. However, the implementation of finite state machines that do not have a planar layout might be challenging. Thus, it is a question for further research if this scheme can be extended easily to the design of finite state automata, whose layout is non-planar. The molecular computer for logical control of RNA expression can be useful in medical field if it can be used inside a cell, and the programmable walkers can be a really useful tool in nanopartical transportation systems at nanoscale. In conclusion, the designs provided in this chapter might provide

useful insight for research into many interesting problems in nanotechnology.

Chapter 6

A DNA Nanotransport Device Powered by Polymerase $\phi 29$

Polymerase is an enzyme responsible for replication of DNA-RNA template and hence sustenance of life processes. In this chapter, we present a method to exploit a strand-displacing polymerase $\phi 29$ as a driving force for nanoscale transportation devices. The principle idea behind the device is strong strand displacement ability of $\phi 29$, which can displace any DNA strand from a template, while extending a primer on the template. This capability of $\phi 29$ is used to power the movement of a target nanostructure on the DNA track. The major advantage of using a polymerase driven nanotransportation device as compared to other existing nanorobotical devices is its speed. $\phi 29$ polymerase can travel at the rate of 2000 nucleotides per minute at room temperature, which translates to approximately 680 nanometers per minute on a nanostructure. We also demonstrate transportation of a DNA cargo on a DNA track with the help of fluorescence resonance electron transfer (FRET) data.

6.1 Introduction

6.1.1 DNA Nanorobotics

In recent past, there have been tremendous progress in DNA based nanodevices [10, 103, 46, 45, 102, 110, 175, 119, 133, 135, 176, 203, 206, 208]. Recent research has explored DNA as a material for self-assembly of nanoscale objects [41, 94, 109, 159, 195, 201, 202], for performing computation [3, 22, 20, 21, 105, 104, 108, 189, 190, 196], and for the construction of nanomechanical devices [10, 45, 46, 58, 102, 110, 175, 133, 160, 161, 162, 163, 179, 178, 203, 209, 210]. A potential application of autonomous DNA nanorobotical

devices is in the design of a controllable moving device integrated into a DNA lattice for efficient transportation of nanoparticles.

6.1.2 Polymerase as a Machine

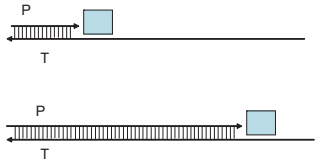


Figure 6.1: A schematic showing a template, a primer, and a polymerase that extends the primer using the nucleotides available in the solution.

We have known polymerase as an enzyme responsible for replication of DNA-RNA template and hence sustenance of life processes. Polymerase helps in replication of DNA-RNA by extending a primer attached to the template by adding the available free complementary nucleotides to its 3' end. Figure 6.1 a) shows a cartoon of a polymerase extending a primer P on a template T, using the bases A, C, T and G from the solution, and adding the complementary nucleotides at the 3' end of the primer P.

There has been active interest of researchers in understanding the actual mechanism of polymerase for extension of primer, and the mechanical properties related to primer extension. Gelles et al [68] reviewed RNA polymerase movements during transcription and studied mechanisms of RNA polymerase translocation along DNA. Wang et al [184] measured force and velocity for single molecules of RNA polymerase. Many researchers preferred to view the polymerase as a machine, and studied the mechanisms of their movements. Most notably, Spirin [170] considered the structure and functions of RNA in terms of a conveying molecular machine. He studied the principal scheme of forward movement of RNA polymerase along the DNA template. Binding of substrates and subsequent energy from chemical reactions, provides successive selection and fixation for next conformational states of enzyme complex. This in turn provides directionality by means of

"Brownian ratchet mechanism". Goel [71] revealed through a series of single-molecule experiments that mechanical tension on DNA can control both the speed and direction of the DNA polymerase motor, and proposed a theoretical description of this tension-induced "tuning" and "switching". Thomen et al [173] addressed the issue of how the enzyme converts chemical energy into motion.

In these experiments, mechanical properties of various polymerase were explored. However, none of them tried to exploit the mechanical energy of the polymerase to transport other objects.

6.2 Our Polymerase Driven Nanotransportation Device

In this chapter, we present the first design of a nanotransportation device powered by a polymerase. We use $\phi 29$, a polymerase known for its exceptional strand displacement activity, to push a DNA cargo. Researchers have studied the structure of $\phi 29$ polymerase and have provided useful insights into its exceptional strand displacement and processivity, and have deduced its translocation mechanism [24, 84, 85, 142].

In Section 6.2.1 we describe the basic principle of our nanotransportation device, and in Section 6.2.2 we describe a high-level design of the device. In Section 6.3, we outline experimental materials and methods. In Section 6.4, we discuss our experimental results in detail.

6.2.1 Basic Principle of our $\phi 29$ Nanotransportation Device

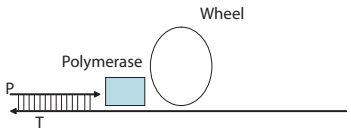


Figure 6.2: Polymerase pushes a wheel on a DNA track

Our polymerase driven nanotransportation device is aimed at exploiting the mechanical energy of polymerase motion, when it travels towards 3' end. Another DNA strand (DNA cargo), when attached to the template blocking the path of polymerase, is pushed by the moving polymerase. Polymerase $\phi 29$ is our choice for pushing the cargo. Figure 6.2 illustrates the basic idea of our $\phi 29$ polymerase nanotransportation device.

In order to brake this polymerase nanotransportation device at a desired destination, we use a sequence of consecutive As (known as *stopping sequence*) on the template. The template does not contain any A in its sequence till the stopping sequence. If the reaction solution lacks the nucleotide T, then the polymerase can still extend the primer till the beginning of stopping sequence, but after that it encounters the sequence of As in the template. Since dNTP T is missing in the reaction solution, the polymerase can not further advance from there, and hence the nanotransportation device gets braked.

The major advantage of using a polymerase driven motor over other nanorobotical devices is its speed. $\phi 29$ polymerase can travel at the rate of 2000 nucleotides per minute at room temperature, which is equivalent to approximately 1400 nanometers per minute on a nanostructure.

6.2.2 Design of $\phi 29$ Polymerase Nanotransportation Device

Figure 6.3 illustrates the basic design of our polymerase based nanotransportation device. The polymerase pushes the wheel on the track (template). It should be noted that the wheel does not roll on the track. It just gets pushed without rolling. The wheel has a 21 bases long complementary sequence to the region of track only near its initial position. Therefore it hybridizes with the template track only at the initial position and nowhere after that. It is needed to ensure that the wheel is attached to the track initially at a unique position. Due to this the wheel does not roll but only slips on the track. However, once the wheel has been displaced from its initial position, it can just slip on the track arbitrarily even without

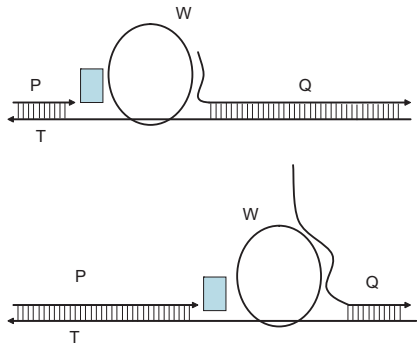


Figure 6.3: Basic design of the polymerase driven nanotransportation device. Protector strand Q prevents the wheel from moving on its own, but is dislodged by polymerase extension of primer P from left.

a push from polymerase. In order to prevent the wheel from slipping away on the track on its own, a strand Q, referred to as *protector strand*, is hybridized to the track. It is shown in Figure 6.3. Another purpose of strand Q is to impart rigidity to the track, which otherwise might fold onto itself.

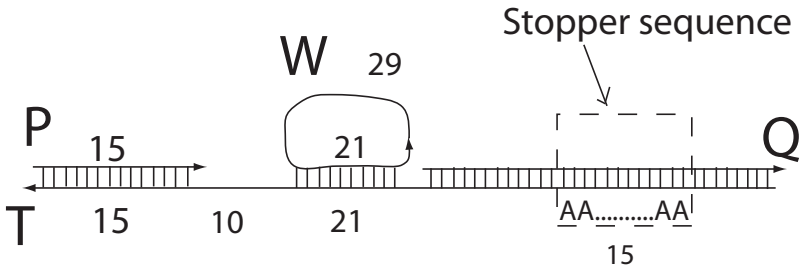


Figure 6.4: The design of polymerase based nanotransportation device in terms of lengths of DNA sequences

Figure 6.4 shows a more detailed design of the system. The track is chosen to be approximately 100 bases long DNA strand. The wheel hybridizes with track in a 21 bases long region, which is 25 bases away from the 5' end of the track, as shown in Figure 6.4. The strand P is a 15 bases long primer complimentary to the first 15 bases of the track T. A free space of 10 bases is left for the polymerase. There is a sequence of 15 consecutive As in the track T, that act as the stopping sequence. The total length of the wheel strand is

50 bases and the protector strand Q is 45 bases long.

We would like to point out a few design constraints, before we describe the experimental methods in Section 6.3. There should not be any As in the track between the initial position of the polymerase and the stopping sequence, so that the polymerase does not stop before the desired position. The primer needs to be more than 6 bases for polymerase $\phi 29$ to work. For the circularization of a single strand DNA (for constructing the wheel), the length greater than 40 bases is preferred. For the polymerase $\phi 29$ the recommended temperature is $30°C$. At $25°C$, there is a 5% loss in efficiency, and there is a 50% - 75% loss in efficiency at $16°C$. Protector strand Q should have di-deoxynucleotide (ddNTP) at 3' end in order to prevent its extension by polymerase $\phi 29$.

6.3 Materials and Methods

6.3.1 Overview of Experiments

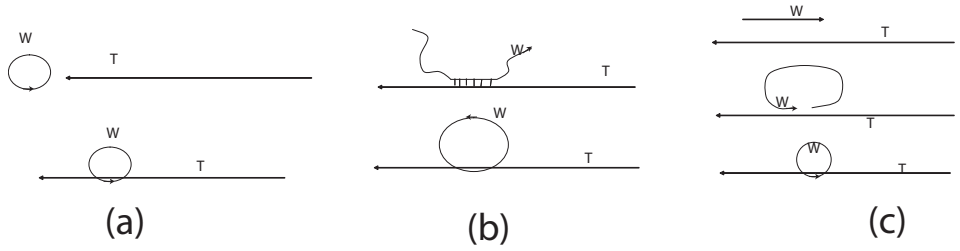


Figure 6.5: Three methods for circularization of wheel: (a) Circularizing the wheel first, and then attempting to hybridize it with track. It is challenging because the track needs to thread through the wheel (b) Partially hybridizing the wheel with the track first, and then circularizing the wheel (c) Padlock probes technique

The very first challenge is to assemble the circular wheel strand on a linear track strand. 5' end of the wheel needs to be phosphorylated so that it can be ligated with its 3' end and form a complete circle. In case the wheel is already circularized, it requires the threading of the track through the wheel to form a double helical region as shown in Figure 6.5 a). It is a challenging condition to meet, and therefore alternative techniques are required. Figure 6.5

b) and c) show two such techniques. In Figure 6.5 b), first the wheel strand is partially hybridized with the track with both its ends free. Then the two ends of wheel are ligated together by using a circularizing technique. However, technique shown in Figure 6.5 c), known as padlock probes [120, 17], is superior to it as the track acts as a linker, and makes the circularization easy. We use the padlock probe technique to attach circular wheel to the track.

In order to ensure that the wheel is always attached to the track, we circularize the track as well by ligating its two ends together. Then the circular wheel is intertwined with the circularized track, and hence does not detach from it. The fact that the track and the wheel are inseparable from each other, makes it easier for us to detect the assembly in a denaturing gel. It also ensures that the wheel stays on the track during the experiment. The phosphate group at 5' end of the track is needed for its circularization. In order to save on a linker strand during circularization of the track, initially we used Circligase enzyme, but it was not very successful as we will discuss in Section 6.4. Therefore, we modified the design slightly, and use a strand BP as a linker-cum-primer for our nanotransportation device. Figure 6.3.1 summarizes the entire process. Track strand T is first circularized using the linker-cum-primer strand BP, and then ligated using $T4$ ligase. In the next step, the wheel strand is circularized using the track strand T as the linker, as shown in Figure 6.3.1. This is done by hybridization of the strand W with circularized T, followed by its ligation to seal the nick in it. It should be noted that the presence of the phosphate groups at 5' ends is responsible for these circularizations to work.

Next step is the hybridization of protector strand Q onto this assembly. It should be noted that ddNTP (dideoxy NTP) is required at the end of strand Q to prevent it from extending under the influence of polymerase. As mentioned earlier, we leave a space of 10 bases for polymerase $\phi 29$ between the strand BP and the wheel on the track. The wheel is chosen to be 50 bases so that it can be easily circularized. The track contains

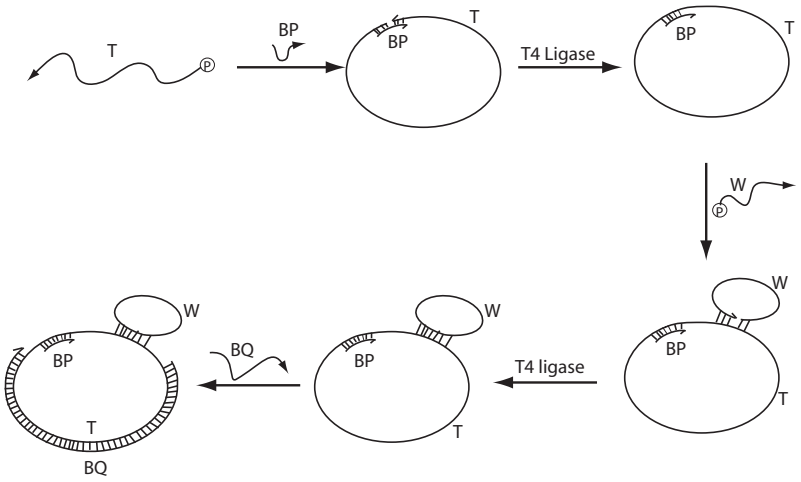


Figure 6.6: Overview of the complete set up assembly of polymerase based nanotransportation device

15 consecutive As as the stopping sequence. It is expected that in presence of all the nucleotides in the reaction solution, the polymerase $\phi 29$ will keep extending the primer and circling on the circular track, while displacing any strand that comes in its way. This results in a rolling circle amplification, as described in Section 6.4.2.

6.3.2 Experimental Details

DNA sequences for the polymerase motor, denoted as T, W, Q, P, BP, and BQ were designed and optimized with the SEQUIN software[157]. T is a $97mer$, W is a $50mer$, Q is $45mer$, P is $15mer$, BP is a $25mer$, and BQ is 35 bases long. In the strand T and W, 5' end is phosphorylated to circularize them using $T4$ ligase. Table 6.1 shows the various DNA sequences used. DNA strands were synthesized commercially by Integrated DNA Technologies, Coralville, IA, and purified by denaturing gel electrophoresis. DNA stock solutions were prepared at a concentration of 30μM in ultra pure water. Concentrations of DNA strands were determined from the measurement of ultraviolet absorbance at 260 nm.

TAE/Mg buffer (1X buffer: 0.04 M Tris acetate, 1mM EDTA, 12.5 mM Mg acetate, pH 8.3) was used for reactions including annealing, formation of native gels, and running

name	symbol	sequence
Track	T	/5Phos/AAT CAC CAT AGT GCA ACC TGA AAA AAA AAA AAA AAT GTG CCT CTG TTC TGC TCG CTT GCT GCG TTG GCT GTC GTG TCC TTG TTA CTA AGA TGC TTA C
Wheel	W	/5Phos/AGC GAG CAG AAA AAA AAA AAA AAA AAA AAA AAA AAA AAA CCA ACG CAG CA
Protector	Q	CAG AGG CAC ATT TTT TTT TTT TTT TCA GGT TGC ACT ATG GTG ATT
Primer	P	GTAAGCATCTTAGTA
Protector	BQ	CAG AGG CAC ATT TTT TTT TTT TTT TCA GGT TGC AC
Primer-linker	BP	TAT GGT GAT TGT AAG CAT CTT AGT A
	tT18	TGT GGA CCG TAA ATG /iSp18/ACG ATT CCA GCG AGC GAA CGT G
	tTD	TGT GGA CCG TAA ATG /idSp/ ACG ATT CCA GCG AGC GAA CGT G
	tT2D	TGT GGA CCG TAA ATG /idSp//idSp/ ACG ATT CCA GCG AGC GAA CGT G
	tP	TCA GAA TTG GCA CGT TCG CTC G

Table 6.1: The sequences used for the demonstration of polymerase driven motor

native gels.

TBE buffer (1X Buffer: 0.089M Tris-base, 0.089M Boric Acid, 0.002M EDTA (disodium), pH 8.3) was used for preparation of denaturing gels and running denaturing gels.

Circligase enzyme from Epicentre Biotechnologies,Madison, WI was used initially for circularizing the DNA strands. It was provided with CircLigase buffer (10X Reaction Buffer: 0.5 M MOPS (pH 7.5), 0.1 M KCl, 50 mM $MgCl_2$ and 10 mM DTT). The reaction buffer does not contain ATP or $MnCl_2$ which must be added to the reaction at final concentrations of 0.05 mM ATP and 2.5 mM $MnCl_2$. The reaction mixture was incubated for 1 hour at $60°C$ for circularizing and then heated at $80°C$ for 10 minutes to deactivate the circligase.

T4 DNA ligase from New England Biolabs was used for ligation. It was provided with T4 ligase buffer (1X buffer: 50 mM Tris-HCl, 10 mM $MgCl_2$, 1 mM ATP, 10 mM Dithiothreitol, pH 7.5 @ $25°C$). Ligation was performed by incubating the sample with

ligase and ligase buffer at $16°C$, and ligase was deactivated by heating the sample at $65°C$.

$\phi 29$ polymerase from New England biolabs was used to power our nanotransportation device. It was provided with $\phi 29$ polymerase buffer (1X buffer: 50 mM Tris-HCl, 10 mM $(NH_4)_2SO_4$, 10 mM $MgCl_2$, 4 mM Dithiothreitol, pH 7.5 @ $25°C$). Polymerase reaction was performed at $30°$ C, in presence of $\phi 29$ polymerase, 1X $\phi 29$ polymerase buffer, $200\mu M$ dNTPs and $(200\mu g/ml)$ BSA (*Bovine Serum Albumin*). The mixture was then heated till $65°C$ to deactivate the polymerase.

Taq polymerase from Invitrogen Inc. was used in comparative studies of braking the nanotransportation device. In addition to, 1X Taq polymerase buffer (20 mM Tris-HCl, 50 mM KCl, pH 8.3 @ $25°C$), $MgCl_2$ at $1.5mM$, dNTPs at $200\mu M$ and Taq polymerase at 25 units/ml are needed for replication by Taq polymerase. The polymerase reaction with Taq polymerase were carried out at $75°C$, and there was no heat inactivation.

Denaturing polyacrylamide gel electrophoresis (PAGE) as well as non-denaturing polyacrylamide gels (acrylamide-bis 19:1) was used to analyze the reaction mixture and verify the products formed. The conformation of the motor was estimated from the size of various participating structures. For denaturing gel electrophoresis, the mixture was heated at $90°C$ for 10 minutes, and then added to denaturing polyacrylamide gel. For imaging, denaturing as well as native gels were stained with $0.5\mu l/ml$ of Ethidium Bromide (EB) solution (from Apex Bioresearch) in 200 ml distilled water for 20-25 minutes. The gel was then viewed under UV transillumination, and images were acquired using an Alpha Imager (Alpha Innotech, San Leonardo, CA). 50 and 100 bp ladders from New England Biolabs were used as references during gel electrophoresis.

6.4 Results and Discussion

6.4.1 Assembly and Demonstration of our Nanotransportation Device

Construction of Circular Track Using Linker Strand

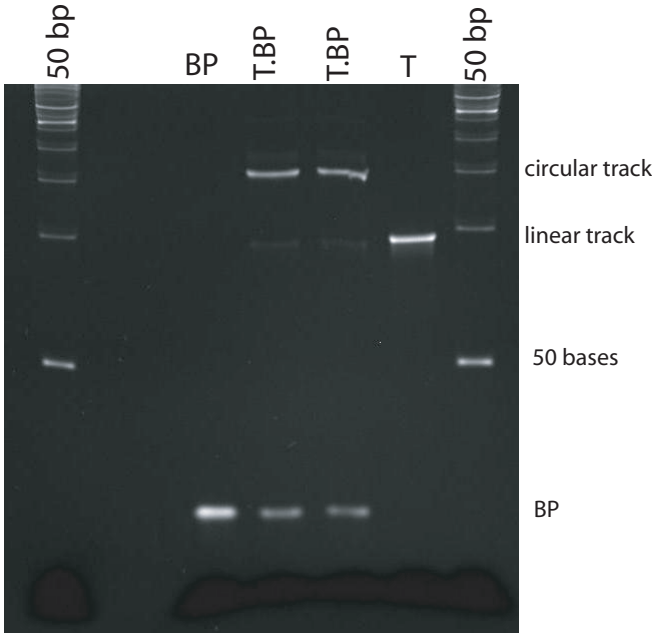


Figure 6.7: Strand T circularized using linker strand BP. The bottom most bands correspond to BP. And it is marked against 50 bp ladder.

The inefficiency of circligase, and the need of subsequent purifications are detrimental in terms of yield and time-consumption. It is also possible that circligase interfered with the T4 ligase. This might be responsible for non-formation of our desired structure with intertwined circular wheel and circular track. Therefore, we tried yet another approach, where we eradicated the need of circligase, and instead used a linker strand to ligate the two ends of the track together to form a circle.

T was circularized using a linker strand BP. First T and BP are annealed, and then $T4$ ligase is added to this $T \cdot BP$ solution in order to ligate the two ends of T together to form a complete loop. Thus, we obtain BP hybridized to the circular track, where it can act as a

primer as well for the later parts of the experiment. Sequences for BP and T are given in the Table 6.1.

10 μl of 30 μM T was mixed with 10 μl of 30 μM BP along with 10 μl of $10X$ TAE buffer. Water was added to make the final total volume 100 μl. The solution was heated till $90°C$ and then cooled down to the room temperature over a period of 4 hrs. $100\mu l$ of $T.BP$ formed was divided into 5 aliquots of $20\mu l$ each. $0.6\mu l$ of $T4$ ligase was added to each of the aliquots, along with $5\mu l$ of 10X ligase buffer with water added to make the solution volume 50 μl. Effectively, $3\mu l$ of $T4$ ligase was added to 100 μl of $T.BP$ complex, along with $25\mu l$ of 10X ligase buffer with water added to make the final solution volume $250\mu l$. The solution was incubated at $16°$ C for 4 hours, and later heated at $65°C$ for 10 minutes to deactivate the ligase enzyme. 5 μl aliquots were taken from the resultant solution to analyze in 10% denaturing gel (run at $50°C$ at 220 V for 1.5 hours). Figure 6.7 shows strands BP and T against $T.BP$ (ligated) in the denaturing gel. In the two wells in the center, the topmost bands correspond to circular track. The strand BP separates from the circular track in denaturing gel and can be seen at the same height as BP in Figure 6.7.

Attachment of Wheel onto the Circular Track

Next strand W is added to the circular track and is circularized using padlock probe method. 270 pmoles of $T.BP$ ($225\mu l$ of $1.2\mu M$) is annealed with $9\mu l$ of 30 μM W. 16 μl of 10X TAE is added to it (as $9\mu l$ was already remaining in the previous solution), and water was added to make the final solution volume $250\mu l$. This solution was annealed for 4 hours (cooling from $80°$ to room temperature). Then the solution was divided in 5 aliquots of 50 μl each. In each of the aliquots, T4 ligase was added with T4 ligase buffer, and water. In total, $5\mu l$ of T4 ligase with $4.8\mu l$ ($21.2\mu l$ ligase buffer was already present in the solution from previous step) of ligase buffer along with $0.2\mu l$ of water added to make the total volume $260\mu l$. The solution was then incubated at $16°C$ for 4 hours and later heated at

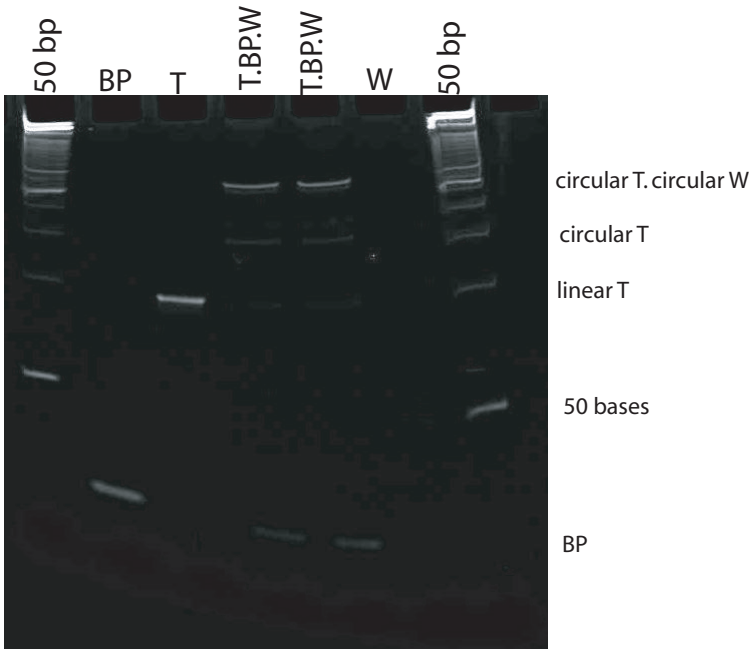


Figure 6.8: Strand W is hybridized with $T.BP$ as shown in Figure 6.3.1, and subsequently ligated to form circular wheel and circular track intertwined with each other. Denaturing gel is unable to separate them

$65°$ to deactivate the ligase. Thus we have the product $T.BP.W(ligated)$ formed with two ends of W ligated with each other.

The product was analyzed using 10% denaturing gel as shown in Figure 6.8. The wheel and the track are intertwined with each other as desired. In Figure 6.8, the topmost bands in wells labeled as $T.BP.W$ are circularized wheel and circularized track intertwined with each other. The wheel is ligated successfully so that they form an structure of two rings intertwined with each other. It is evident from the fact that these two rings are unable to separate from each other in the denaturing gel shown in Figure 6.8. The two vague bands below this band in the $T.BP.W$ well correspond to the circular track and the linear track.

6.4.2 Action of Polymerase $\phi 29$

223.5 μl of $T.BP.W$ solution from previous step was mixed with 7.8 μl of BQ (30 μM) along with $2.7\mu l$ 10X TAE.Mg buffer (21.27 μl 10X TAE was already present present in

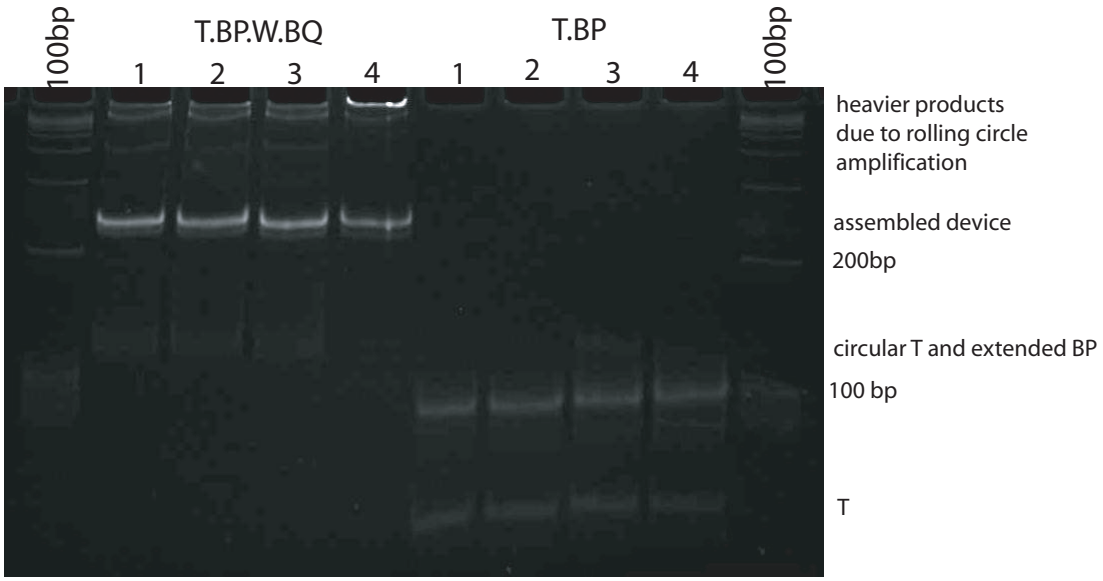


Figure 6.9: Polymerase $\phi 29$ acts on $T.BP.W.BQ$ and $T.BP$ in presence of 1, 2, 3, and 4 dNTPs

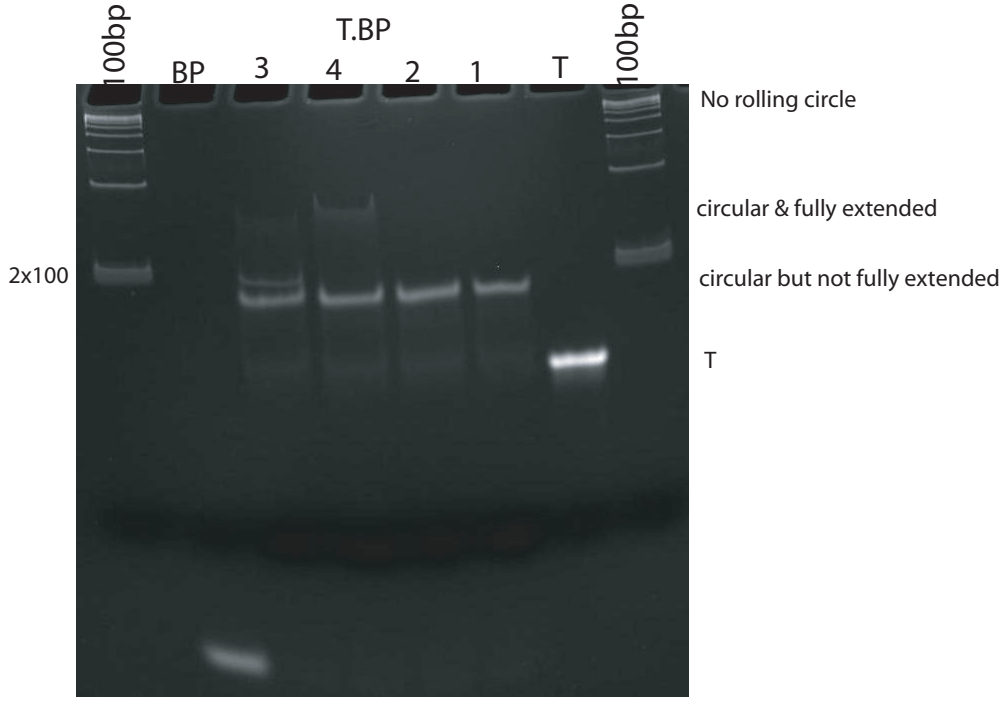


Figure 6.10: Polymerase $\phi 29$ acts on $T.BP$ in presence of 1, 2, 3, and 4 dNTPs

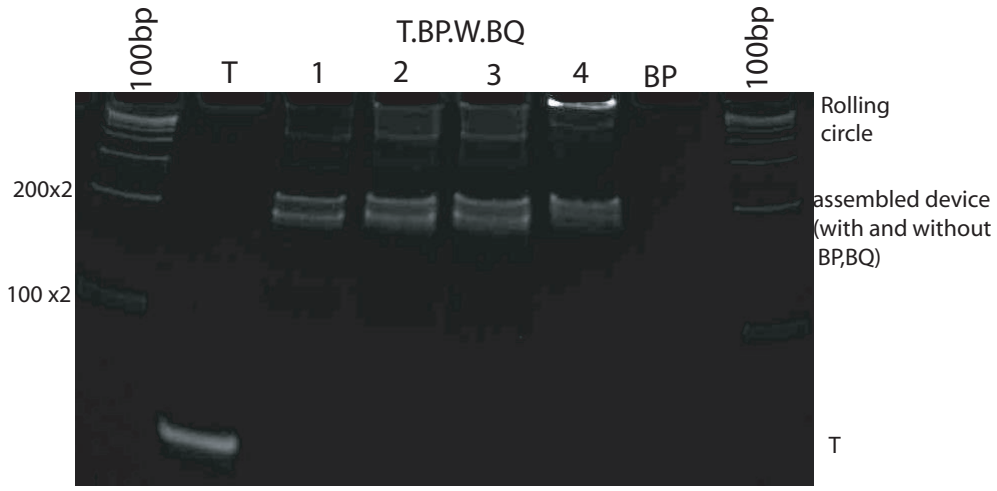


Figure 6.11: Polymerase $\phi 29$ acts on $T.BP.W.BQ$ in presence of 1, 2, 3, and 4 dNTPs

the solution). Deionized water was added to make the final volume 240 μl, and the solution was annealed (heated till $75°C$ and cooled down to room temperature over 2 hours). Multiple samples were drawn from it for various experiments of polymerase $\phi 29$ under different conditions.

First of all, four samples each containing 15 μl of $T.BP.W.BQ$ were prepared. In one sample, 0.4 μl of each of the 4 dNTPs, 0.6 μl BSA (bovine serum albumin), 3 μl of 10X polymerase buffer, and 0.07 μl of ϕ29 polymerase were added. Other samples were similar except that in second sample dNTP T was not added, and in the third sample C and T were not added, and in the fourth sample only nucleotide A was added.

At the same time, we annealed 120 pmoles of T and BP in 60μl total solution in presence of $1X$ TAE.Mg^{++} buffer to form $T.BP$. This was annealed from $90°C$ to room temperature over a period of 2 hours.

Four samples each containing 15 pmoles of $T.BP$ were drawn from it, and $\phi 29$ polymerase with polymerase buffer, BSA and dNTPs were added to them as follows: first sample contained all the dNTPs, second sample contained all but T, third lacked C and T, and the fourth sample had only nucleotide A in it.

These eight samples were analyzed simultaneously in 10% native gel as shown in Figure 6.9. $T.BP.W.BQ$ sample with four nucleotides exhibit the phenomenon of rolling circle amplification, due to the extension of the strand BP on the circular track. The presence of multiple bands in case of $T.BP.W.BQ$ implies the formation of various intermediate products, but the rolling circle product formed on T.BP.W.BQ in presence of four nucleotides and polymerase $\phi 29$ is most dominant. However, one undesirable characteristic of $\phi 29$ that comes to the fore is that in presence of excess phi29 it does not show a good exonuclease activity, and thus does not always stop at the stopping sequence. Thus braking mechanism in our nanotransportation device requires that $\phi 29$ should not be not used in excess, as we discuss in Section 6.4.3.

At the same time, Figure 6.9 also shows that in case of $T.BP$, the longest product formed is approximately 200 bases in weight, and a rolling circle amplification is not observed in $T.BP$. This is due to the nick present in the track T, as $T.BP$ used in this experiment was not ligated to seal the nick. Thus, the nick in the track prevents the $\phi 29$ from extending BP beyond the nick, which is a useful secondary information derived from this experiment.

The wheel as well as track are already ligated to form a circle, the only primer in our setup are the strand BP and BQ. The rolling circle amplification indicates the extension of only these two strands. As we had already shown in Section 6.4.1 that wheel and track formed two circles intertwined with each other (inseparable in a denaturing gel). Therefore, the circular motion of the polymerase on the circular track during the rolling circle amplification should imply that the wheel gets pushed on the track by polymerase, due to the exceptionally strong strand displacement properties of the $\phi 29$ polymerase.

We repeated the above experiment and analyzed the products separately on two native gels as shown in Figure 6.10, and Figure 6.11, with the same conclusions.

6.4.3 Brakes on Polymerase Driven Nanotransportation Device

In Section 6.4.2, we have seen evidence that the wheel is pushed by the polymerase to construct a fast nanoscale motor. However, for imparting more usefulness to this device, it is needed that it should be able to stop at desired location(s). Successful braking is also important to design advanced applications of the system. As described earlier, the stopping mechanism is based on a sequence of 15 consecutive As on the track and the lack of the dNTP T in the reaction mixture. It causes the polymerase to get stuck at the stopping sequence and in effect stops or brakes the nanotransportation device. We performed a series of experiments to test the efficiency of this braking mechanism and to determine the conditions favorable for it.

We tested our braking mechanism against 2 polymerase: $\phi 29$ (used in our nanotransportation device) and Taq (for the sake of comparison).

Braking Capabilities of Taq

This experiment was done to test the effect of spacers and stopping sequence on the motion of polymerase. Sequence tP (Table 6.1) is used as a primer and $tT18$, tTD, and $tT2D$ (Table 6.1) are used as templates. $tT18$ contains Spacer 18 which is an 18-atom hexa-ethyleneglycol spacer. It is the longest spacer arm that can be added as a single modification. On the other hand tTD, and $tT2D$ contain d-Spacers (1',2'-dideoxyribose (dSpacer)), which are used to introduce a stable abasic site within an oligonucleotide. The sequences for $tT18$, tTD, and $tT2D$ are given in Table 6.1.

5 μl of 30 μM template ($tT18$, tTD, $tT2D$ respectively) was mixed with 5 μl of 30 μM primer (tP). 10 μl of $10X$ Taq polymerase buffer, $50mM MgCl_2$, 0.4 μl Taq polymerase, and $2\mu l$ of each 10mM dNTP were added to each of the 3 samples. For each of the template 3 different versions were prepared (with no dNTP, with all dNTPs but T, and with all the four dNTPs). Deionized water was added in each of the 9 samples to make

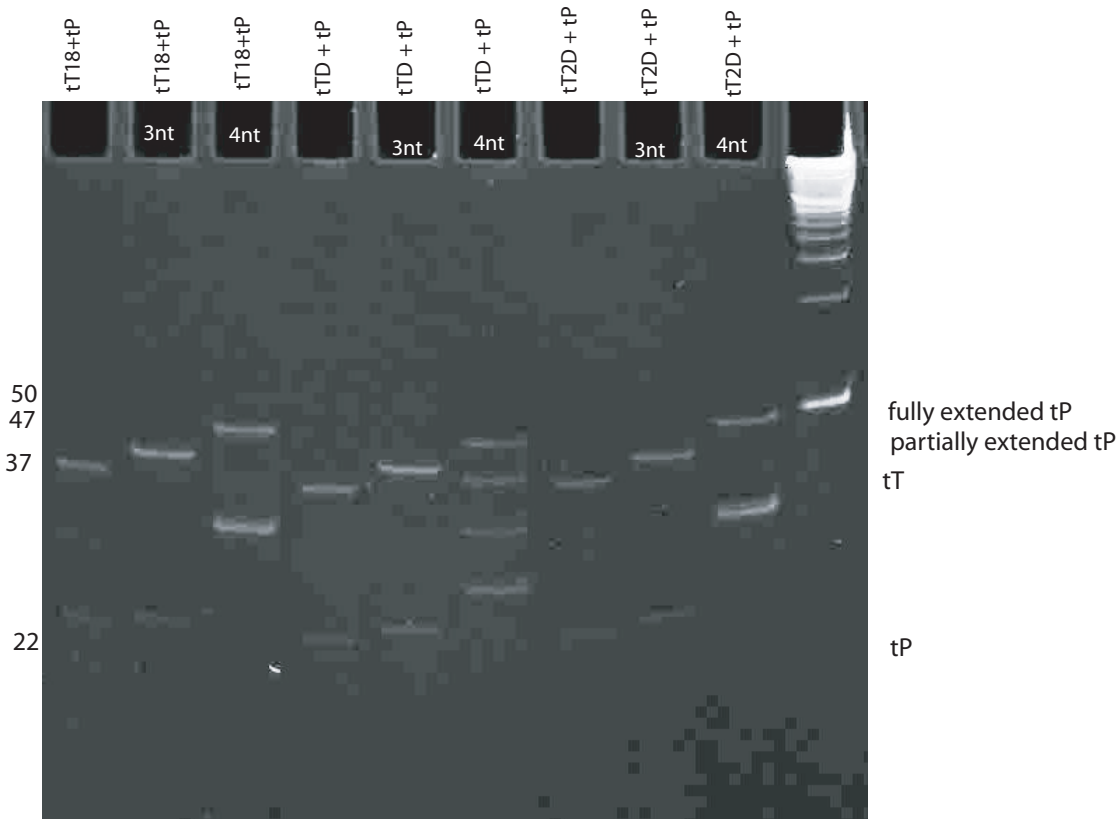


Figure 6.12: Braking capabilities of a stopping sequence for Taq polymerase. The higher weight product formed in presence of 4 dNTPs as compared to 3 dNTPs indicates good braking for Taq

their individual total volumes $100\mu l$. The solutions were incubated at $75°C$ for 1 hour.

10% denaturing gel in Figure 6.12 shows that a heavier product is formed in case of 4 dNTPs as compared to 3 dNTPs in each of the 3 templates, which indicates that sequence of 3 consecutive As is acting as a brake successfully. In 1st, 4th and 7th well from left the top band corresponds to $tT18$, tTD, and $tT2D$ (37 bases) and the bottom band is tP (22 bases). In case of 4 nucleotides in presence of polymerase each of them form a product that is 47 bases long, which can be seen at the same height as 50 bp marker. However in case of 3 nucleotides they stop at 3 consecutive As, and therefore products of length lesser than 47 but more than 37 are formed and the difference is clearly visible in Figure 6.12. The inability of spacers to prevent the Taq polymerase from extending the primer to maximum

length indicates the inability of spacers to act as brakes and hence, stopping sequence of consecutive As is needed for braking.

Braking Capability of $\phi 29$

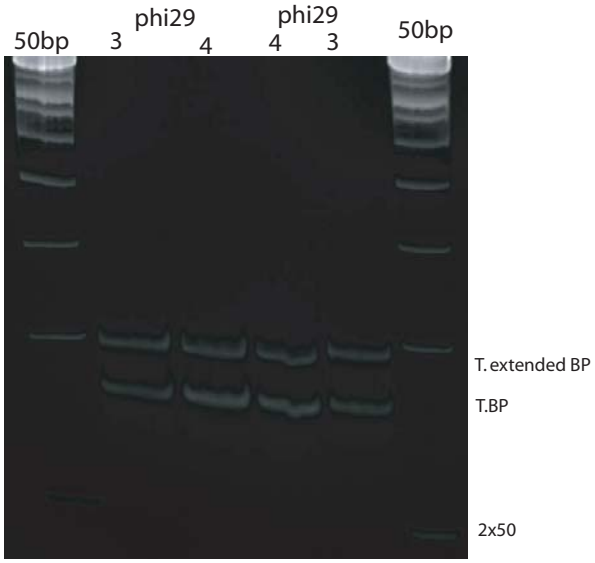


Figure 6.13: Braking capability of a stopping sequence of length 15 is tested in excess polymerase $\phi 29$

We tested the braking of polymerase $\phi 29$ on $T.BP$ (unligated). In this case, the sequence of $15A$s in the track, called as stopping sequence, acts as the brake, when all dNTPs except T are present in the reaction mixture. The sequences for T and BP are provided in Table 6.1.

$T.BP$ was formed by annealing as described earlier. It was not ligated. Then $1\mu l$ $\phi 29$ polymerase was added to 20 μl of $T.BP$ solution along with 2 μl of 10X polymerase buffer, along with dNTP, BSA, water. Two kinds of samples were prepared: with all the 4 dNTPs added and with all dNTPs except T added. The samples were incubated at $30°C$ for 30 minutes, and then polymerase was deactivated by heating to $65°C$ for 10 minutes.

10% native gel in Figure 6.13 shows that $\phi 29$ did not work well with our braking mechanism, when taken in excess. In that case the exonuclease part of the polymerase that

is responsible for proofreading does not work so well. It can be seen from the Figure 6.13 that there is no difference in the product formed with 3 or 4 dNTPs. Therefore, we need to lower the concentration of $\phi 29$ polymerase in subsequent experiments. However, this experiment reconfirmed that $\phi 29$ polymerase is not able to extend the primer beyond a nick. In subsequent experiments, we compared braking abilities of Taq and $\phi 29$ on $T.BP$ with decreasing concentration of $\phi 29$.

Comparative Study for Testing Brakes Using Taq and $\phi 29$

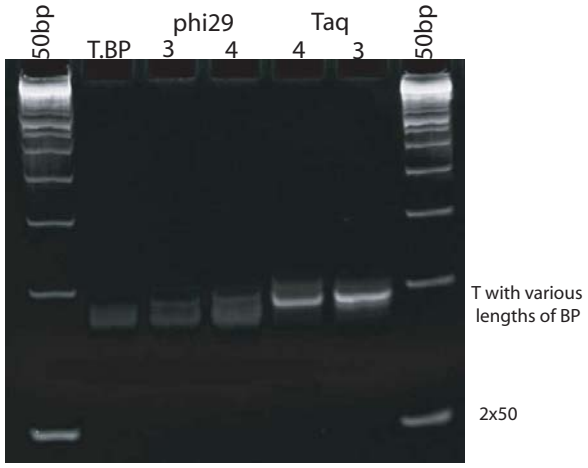


Figure 6.14: Braking the nanotransportation device using $\phi 29$ and Taq

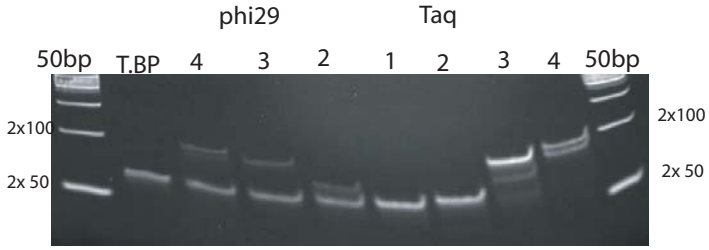


Figure 6.15: Effect of reducing the quantity of $\phi 29$ on braking. Comparative study of braking using $\phi 29$ and Taq

We performed comparative studies on effectiveness of stopping sequence based brakes with regards to Taq and $\phi 29$. Unligated $T.BP$ as described earlier in Section 6.4.3 was

used with Taq and $\phi 29$ polymerase respectively.

In the first experiment, $0.10\mu l$ of $\phi 29$ was used with polymerase buffer, dNTPs, BSA, and water. Two samples were prepared: with 4 dNTPs added and with all dNTPs except T. At the same time 2 more samples (with 4 dNTPs, and 3 dNTPs) were prepared for Taq polymerase. Taq polymerase buffer, dNTP, and water were added to them.

Figure 6.14 shows the 10% native gel for these samples. It can be seen that it is hard to distinguish between the product formed in presence of all 4 dNTPs vs 3 dNTPs in case of $\phi 29$. It completes one full circle whether all 4 nucleotides are present or only 3 nucleotides are present. We repeat that rolling circle amplification is not observed due to the nick. On the other hand, in case of Taq, a slightly higher band can be seen in 4 nucleotides as compared to 3 nucleotides, which indicates good braking.

10% native gel in Figure 6.15 shows a similar experiment with quantity of $\phi 29$ decreased to $0.07\mu l$ in its samples. It can be seen that in $\phi 29$ samples, in presence of 2, 3, or 4 dNTPs, different products are formed, shown by the existence of different bands in the 10% native gel. With 4 nucleotides, it completes one full circle (no rolling circle because of the presence of nick), while with 3 nucleotides, it stop at the stopping sequence, and with 2 nucleotides it stops even earlier. Thus at this concentration of $\phi 29$ braking mechanism works well.

Thus, we can conclude that it is not easy to put good brakes on $\phi 29$. In excess of $\phi 29$, it has a poor exonuclease activity. In such a situation, whenever $\phi 29$ does not find the correct base, it adds incorrect bases to the to extend the primer and moves further. On the other hand Taq was immaculate in braking. It stops if it does not find the correct nucleotide in the reaction solution. The inability of $\phi 29$ to stop at stopping sequence of consecutive As in the absence of T in presence of excess $\phi 29$ is poorly understood, and exploring it is beyond the scope of this work. However, lower concentration of $\phi 29$ is favorable for our braking mechanism as illustrated by Figure 6.14 and 6.15.

6.5 FRET Experiments for Further Verification

The PAGE analysis presented in previous sections presents an indirect method to verify the activity of polymerase based nanotransportation device. In this section, we present another scheme for testing the nanotransportation device in a more direct way. FRET (Fluorescence Resonance Energy Transfer) methods are widely used in the verification of nanomechanical devices.

6.5.1 FRET on our Polymerase Based Nanotransportation Device

In our polymerase based nanotransportation device, if the track is chosen to be linear as opposed to circular for the FRET experiments, then the wheel might disconnect from the track and we might notice the false positives. Therefore we need a circular track. It should also be noted that in this experiment, we need to stop the wheel at the stopping sequence, otherwise its final position might become indeterminable as it can keep doing rolling circle amplification.

In our setting, the simplest design is incorporation of an internal quencher in the circular track, and an internal fluorophore in the wheel to perform fluorescent resonance energy transfer. In the initial position, the distance between the fluorophore and quencher is small and hence fluorescence is perfectly quenched. However the situation changes, when the wheel is pushed by the polymerase. The new distance between fluorophore and quencher is more and hence no quenching is observed. It proves that the wheel moved from the initial position.

However, in order to prove that the wheel reaches the final destination, such a scheme is not sufficient because the relative position of wheel with respect to track is not deterministic at the final destination. The reason is that the wheel does not contain subsequence complementary to any other regions of the track T in order to ensure the initial unique position of wheel.

DNA strands with internal quenchers are extremely costly to be synthesized. 3' or 5' quencher hinders the ligation (and hence circularization) capabilities of a strand. Therefore, internal quencher is mandatory with this approach. But in spite of being costly this approach is not good enough to prove that the wheel reaches the desired final destination.

6.5.2 Design for FRET Experiments

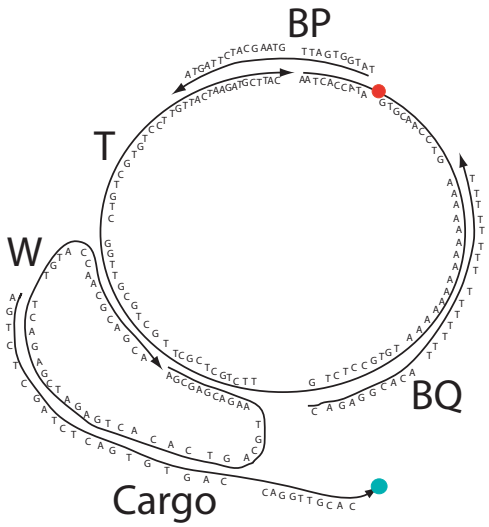


Figure 6.16: Shows the detailed sequence design for the fluorescence experiment

In view of the limitations described in Section 6.5.1, we present a new design of polymerase based nanotransportation device for FRET experiments as illustrated in Figure 6.16. The basic idea is that the wheel carries a cargo having the quencher on one of its end, and different positions are chosen for fluorophore in order to prove the following:

1. The first thing we need to verify is that the cargo is never dislodged from the wheel. Not even by polymerase. This can be demonstrated by attaching a quencher in the cargo at 5' end, and a fluorophore in the wheel at the corresponding position. All fluorescence should be quenched, and even in the presence of polymerase the fluorescence should not appear. This ensures that the cargo, in no conditions, is dislodged

symbol	sequence
PM3.T:	/5Phos/AAT CAC CAT AGT GCA ACC TGA AAA AAA AAA AAA AAT GTG CCT CTG TTC TGC TCG CTT GCT GCG TTG GCT GTC GTG TCC TTG TTA CTA AGA TGC TTA C
PM3.W:	/5Phos/ AGC GAG CAG AAT GCA GTC ACA CTG AGA TCG AGA CTT GTA CCA ACG CAG CA
PM3.Cargo:	/5IAbRQ/AGT CTC GAT CTC AGT GTG ACC AGG TTG CAC
PM3.BQ	CAG AGG CAC ATT TTT TTT TTT TTT T
PM3.BP	TAT GGT GAT TGT AAG CAT CTT AGT A
PM2.Track	/5Phos/ T GTG CCT CTG TTC TGC TCG CTT GCT GCG TTG G /iCy5/ CT GTC GTG TCC TTG TTA CTA AGA TGC TTA CAAT CAC CAT AGT GCA ACC TGA AAA AAA AAA AAA AA
PM2.Wheel	/5Phos/ AGC GAG CAG AAT GCA GTC ACA CTG AGA TCG AGA CTT GTA CCA ACG CAG CA
PM2.Cargo	/5IAbRQ/ TACA AGT CTC GAT CTC AGT GTG ACC AGG TTG CAC
PM2.BQ	CAG AGG CAC ATT TTT TTT TTT TTT T
PM2.BP	TAT GGT GAT TGT AAG CAT CTT AGT A
PM1.T	/5Phos/AAT CAC CAT A /iCy5/ GT GCA ACC TGA AAA AAA AAA AAA AAT GTG CCT CTG TTC TGC TCG CTT GCT GCG TTG GCT GTC GTG TCC TTG TTA CTA AGA TGC TTA C
PM1.W:	/5Phos/ AGC GAG CAG AAT GCA GTC ACA CTG AGA TCG AGA CTT GTA CCA ACG CAG CA
PM1.Cargo:	/5IAbRQ/ AGT CTC GAT CTC AGT GTG ACC AGG TTG CAC
PM1.BQ	CAG AGG CAC ATT TTT TTT TTT TTT T
PM1.BP	TAT GGT GAT TGT AAG CAT CTT AGT A

Table 6.2: DNA sequences for FRET experiments for polymerase based nanotransportation device

from the wheel.

2. The next thing we need to verify is that the wheel is moved from its initial position. The 5' end of cargo contains quencher, and the base of track T near this quencher should contain a fluorophore. Then in the absence of polymerase, we should never see the fluorescence signal, and as soon as polymerase is added to the solution we should observe fluorescence signal. Since we have already proved that the cargo is never separated from wheel, this can mean only one thing that wheel has also moved away from the initial point in track.

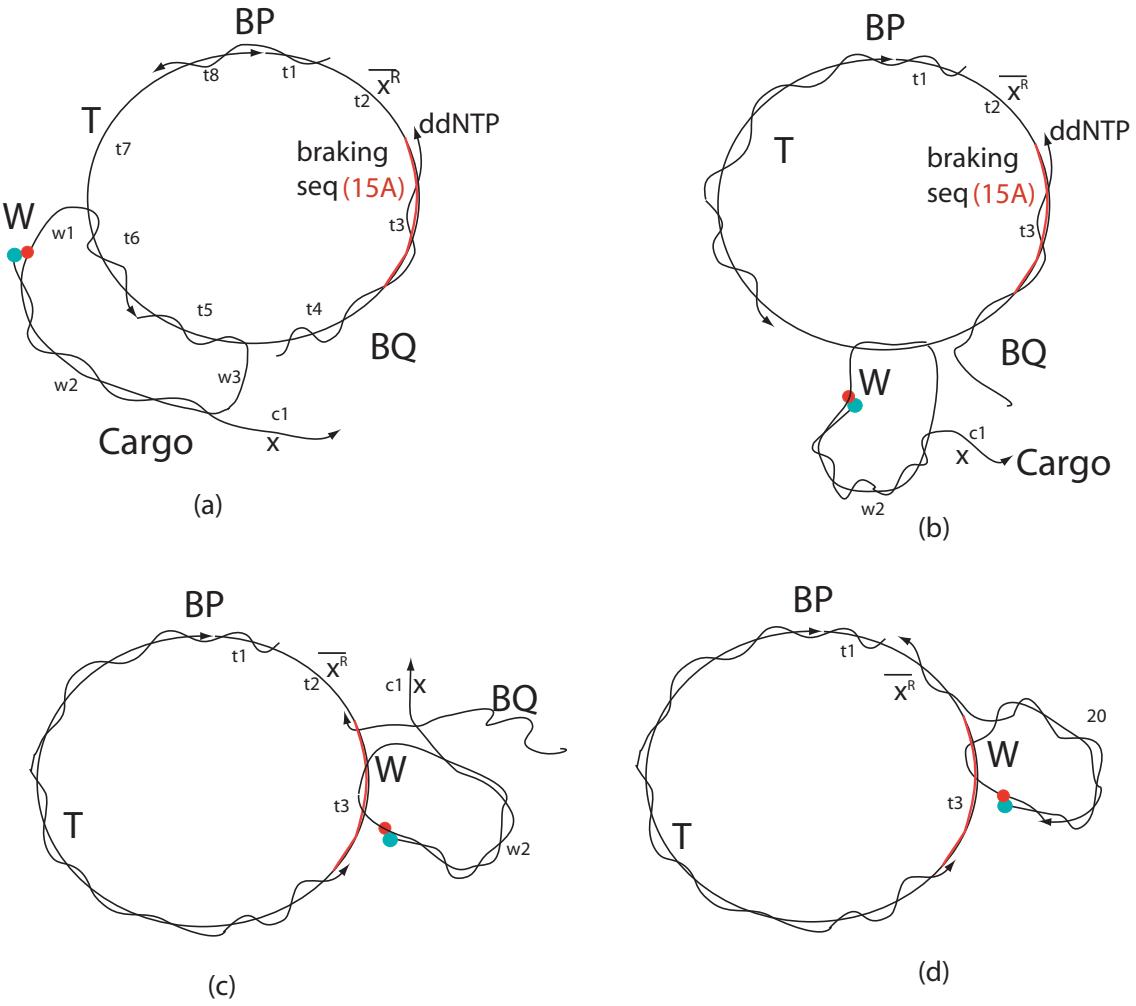
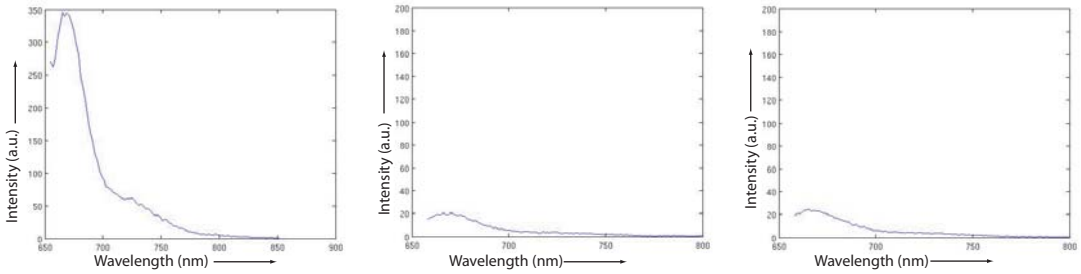


Figure 6.17: Fluorescence experiment that shows that the cargo is not dislodged from the wheel W. The lengths of the various regions of the strands that we used for our experiments are $t_1 = 10$, $t_2 = 10$, $t_3 = 15$, $t_4 = 10$, $t_5 = 11$, $t_6 = 11$, $t_7 = 15$, $t_8 = 15$, $w_1 = 4$, $w_2 = 20$, $w_3 = 4$, and $c_1 = 10$

3. The most challenging part is to prove that the wheel reaches the desired destination. A fluorophore can be inserted in the track at the destination, and 3' end of the cargo contains the quencher. The unhybridized part of the cargo is deliberately chosen to be complementary to the track near the destination region. In absence of polymerase we should observe fluorescence. As soon as polymerase is added, we know from previous two experiments that cargo is never separated from the wheel, and that the wheel moves from the initial position along with cargo, then quenching of the



(a) (b) (c)

Figure 6.18: (a) The fluorescence shown by the assembly in absence of the cargo containing the quencher (b) The fluorescence quenched by the assembly of cargo containing the quencher (c) The fluorescence remains quenched even after the activity of the polymerase $\phi 29$, which indicates that the cargo is not dislodged from the wheel W

signal implies that cargo (along with wheel) has reached the point B. Since the region between the wheel's initial position and destination is double stranded and rigid. All this together implies that wheel reached the destination.

The fluorophore we have chosen is Internal Cy5 with absorbance peak at 648 nm and emission peak at 668 nm, and the quencher is Iowa black RQ with range of 500-700 nm with peak at 667 nm for 3' and 656nm for 5' and is usually recommended for Cy5.

The sequences required for the 3 experiments outlined above are given in the Table 6.2.

In the subsequent subsections, we describe these 3 individual experiments. In each of the experiments, the fluorophores are excited by wavelength 648 nm, and the subsequent emission over all wavelengths (upto approximately 800 nm) has been measured.

6.5.3 Part 1: Demonstration that the Cargo was not Dislodged from the Wheel

In order to prove that the cargo is not dislodged from the wheel in the process, we have a fluorophore on the wheel and a quencher at the end of cargo as shown in Figure 6.17. In case, the quenching disappears (fluorescence appears) during the extension of primer by polymerase, we can conclude that the cargo is dislodged from the wheel. The complete

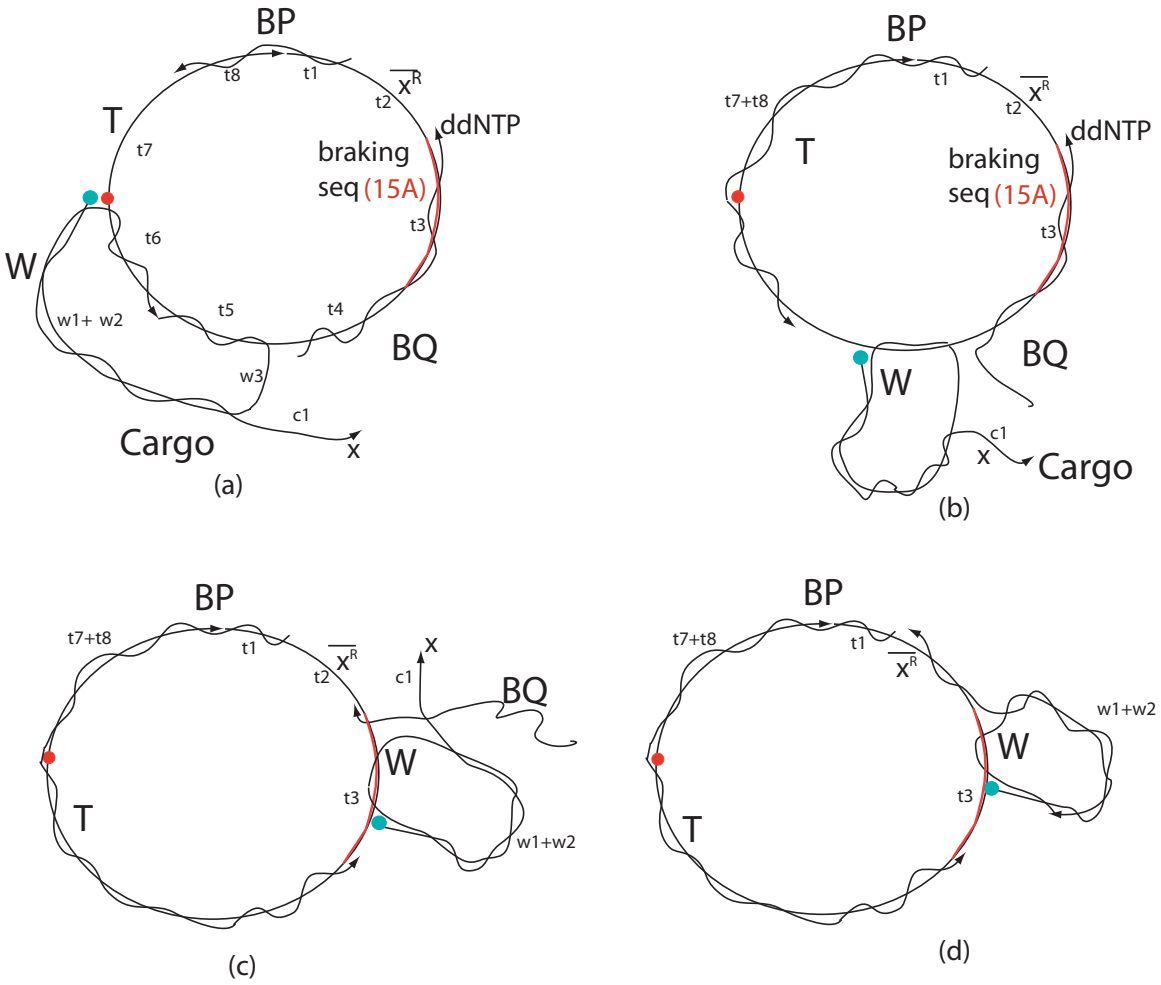
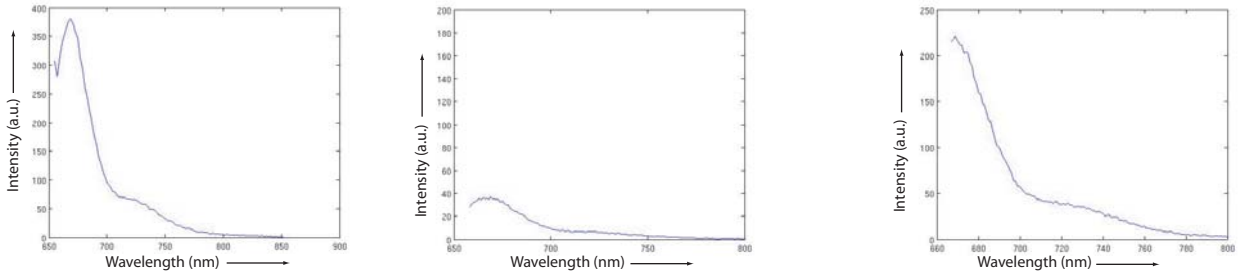


Figure 6.19: Fluorescence experiment that shows that the wheel indeed moved from its initial position. The lengths of the various regions of the strands that we used for our experiments are $t_1 = 10$, $t_2 = 10$, $t_3 = 15$, $t_4 = 10$, $t_5 = 11$, $t_6 = 11$, $t_7 = 15$, $t_8 = 15$, $w_1 = 4$, $w_2 = 20$, $w_3 = 4$, and $c_1 = 10$

sequence for this part are listed in Table 6.2

Initially, the complete device except the cargo (containing the quencher) is assembled, and the fluorescence is measured. Figure 6.18 a) shows the fluorescence in the assembly in absence of the cargo. The cargo is then assembled onto the wheel resulting in the structure shown in Figure 6.17. The fluorescence measurement of the assembled structure is shown in Figure 6.18 b). After the extension of the primer by polymerase $\phi 29$, the fluorescence is measured again (Figure 6.18 c)). The fact that it still shows no fluorescence indicates that



(a) (b) (c)

Figure 6.20: (a) The fluorescence is shown by the assembly in absence of the cargo containing the quencher (b) The fluorescence is quenched after the assembly of the cargo containing the quencher (c) The fluorescence reappears after the polymerase $\phi 29$ pushes the wheel containing the quencher

the cargo is not dislodged from the wheel.

6.5.4 Part 2: Demonstration that the Wheel was Pushed from Initial Position

It is difficult to synthesize the oligonucleotides with internal fluorophore too far from the 5' end. (IDT DNA refused to synthesize such strands), and therefore we did minor redesigning of the strands.

Figure 6.19 shows the entire procedure. The sequences are shown in Table 6.2. The 5' end of cargo contains the quencher, and track has $\backslash iCy5\backslash$ fluorophore at the 32nd nucleotide. The position of the internal fluorophore is 32nd base. The cargo strand in this experiment is 34 bases long instead of 30, with an additional complementary fragment added at the 5' side tailored for this experiment. Iowa Black RQ quencher is attached to 5' end. The complete sequences are listed in Table 6.2.

Initially, the complete assembly except the cargo is constructed, and the fluorescence measurement is taken (Figure 6.20) a). On the assembly of the cargo onto the wheel, the fluorescence is quenched as shown in Figure 6.20 b). But after the extension of primer by polymerase $\phi 29$, the fluorescence can be observed again as shown in Figure 6.20 c). This

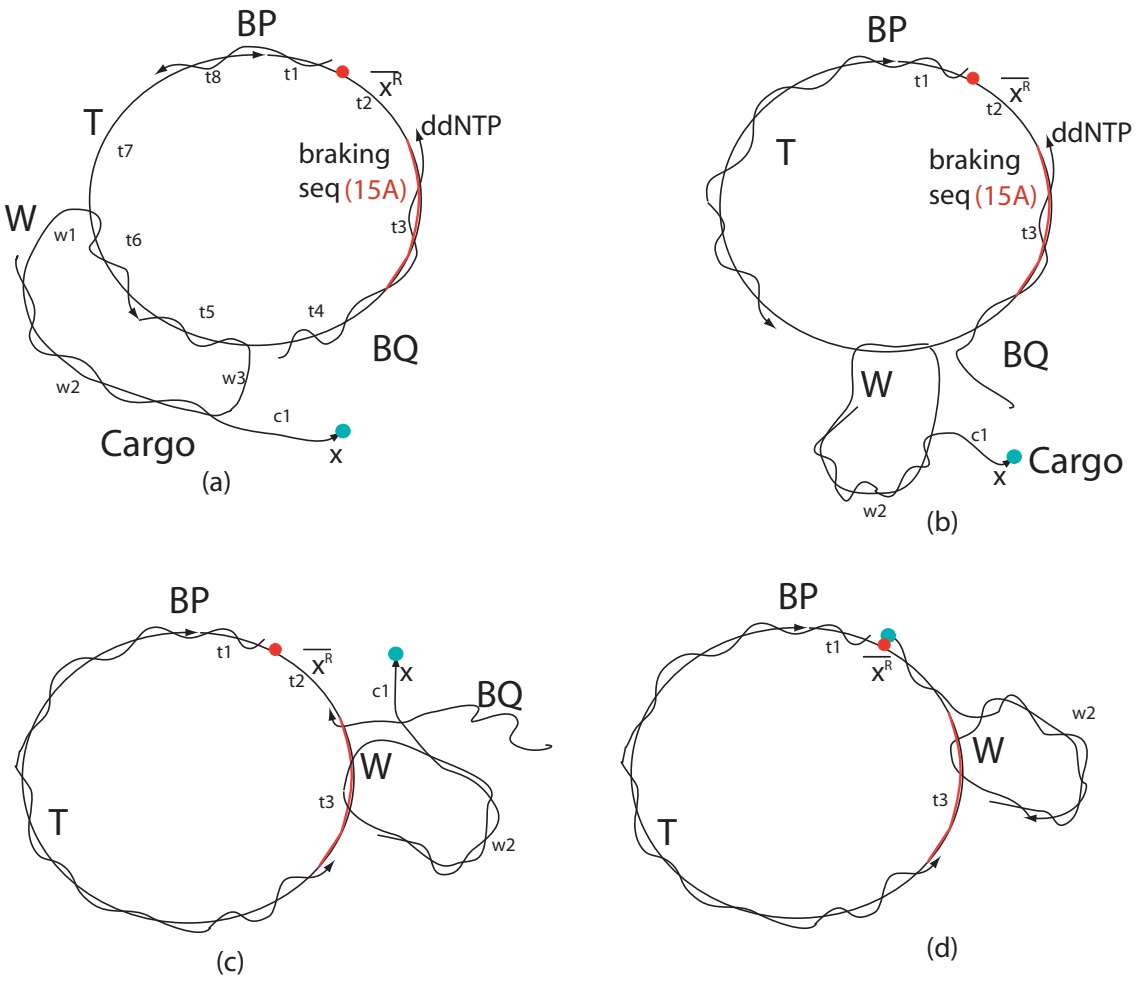
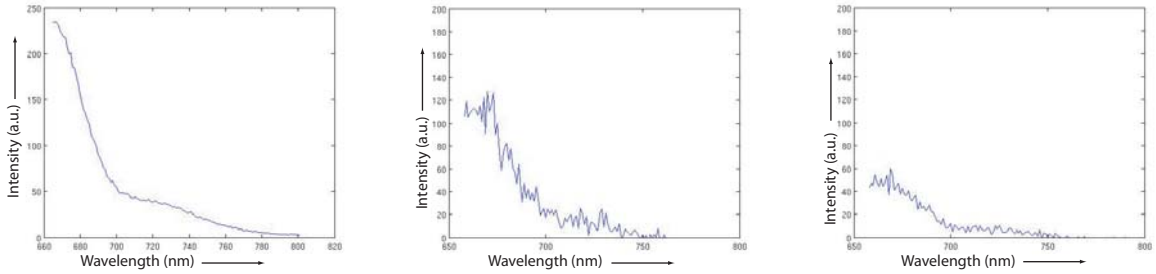


Figure 6.21: Figure shows the setup and step-by-step progress in the fluorescence experiment to demonstrate that the cargo reached the final destination. The lengths of the various regions of the strands that we used for our experiments are $t_1 = 10$, $t_2 = 10$, $t_3 = 15$, $t_4 = 10$, $t_5 = 11$, $t_6 = 11$, $t_7 = 15$, $t_8 = 15$, $w_1 = 4$, $w_2 = 20$, $w_3 = 4$, and $c_1 = 10$

indicates that the cargo is now not close to the fluorophore. We have already shown in the previous section that cargo is not dislodged from the wheel, therefore, it means that wheel is not close to the fluorophore anymore. This implies that the wheel is indeed pushed from its initial position.



(a) (b) (c)

Figure 6.22: (a) The fluorescence is shown by the assembly in absence of the cargo containing the quencher (b) The fluorescence remains after the assembly of the cargo containing the quencher, away from the fluorophore (c) The fluorescence quenches after the polymerase $\phi 29$ pushes the wheel before it stops at stopping sequence, and the sticky end of the cargo hybridizes with the track to quench the fluorescence

6.5.5 Part 3: Demonstration that the Wheel Reached the Desired Final Position

Figure 6.21 illustrates the entire procedure. The sequences are shown in Table 6.2. The quencher Iowa Black RQ is incorporated at the 3' end of the cargo, which is a 30mer, and track has $\backslash iCy5\backslash$ fluorophore at the 11th nucleotide. The difference in the design is because of difficulties in synthesis of oligonucleotides with /iCy5/ away from 5' end.

Initially, the complete device (Figure 6.21) without the cargo is assembled. As expected, the fluorescence is observed as shown in Figure 6.22 a).

Then, low temperature annealing (heated to $45°C$ and then cooled) is performed to assemble the cargo on the track, without the removal of PM1.BQ from the track. Even now, the fluorescence is present, albeit reduced (Figure 6.22 b)).

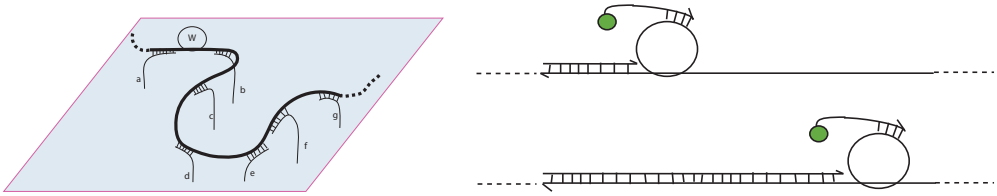
However, once the polymerase $\phi 29$ is added to the solution, and the primer BP is extended, the fluorescence is quenched (Figure 6.22 c)). This indicated that the wheel reached the desired final destination.

However, it should be noted that the assembly of cargo (containing the quencher) resulted in reduction of some fluorescence (Figure 6.22 a) to b)). This is because of the

hybridization of the sticky end, x, of cargo with the $\bar{x}$ subsequence of the track T. One of the purpose of PM1.BQ was to provide rigidity to the track in order to prevent this from happening, but it does not seem to be foolproof. The problem in using our earlier version of BQ is that it will protect the $\bar{x}$ part of sequence permanently, and hence, it might not be available to the cargo at the end.

6.6 Discussion and Future Work

We demonstrated the functioning of a promising nanoscale motor device. The main advantage of using a polymerase driven motor is its speed. As compared to other existing molecular motors based on ligation-restriction[208, 18], dnazymes[176, 46, 175] and fuel-strands[161, 160, 162, 163, 179, 210, 209, 178], a polymerase driven nanotransportation device is much faster. The more popular Taq polymerase is unfit for such an application because of the lack of significant strand displacement activity in it. However we found that $\phi 29$ polymerase does not show good exonuclease activity when present in excess, which causes low fidelity. We also found that $\phi 29$ does not extend a primer across a nick in the template.



(a) (b)

Figure 6.23: (a) A schematic shows a programmable arbitrary track laid on top of an addressable 2D nanostructure from DNA origami (gray surface). The dangler strands are shown using thin lines, and they have free ends that protrude out of the nanostructure. The track is shown using a bold line, that partially hybridizes with the dangler strands in a desirable manner. (b) Figure shows the transport of nanoparticle using polymerase driven motor.

An immediate future goal is to demonstrate two-dimensional routing of the polymerase

nanotransportation device. It may be achieved by demonstrating the motion of polymerase powered nanotransportation device on DNA origami[144] and addressable lattices. Two dimensional nanostructures from DNA origami provides the basic platform. They can be conveniently replaced by two dimensional addressable lattices formed using 4x4 tiles [125] for our purpose. Our idea is to implant a series of single stranded DNA stator strands on the two dimensional plane, so that a track can be assembled on top of the stator strands, as illustrated in Figure 6.6.

Thus, a polymerase based nanotransportation device can provide transport between arbitrary points on a two-dimensional nanostructure along arbitrary path.

Furthermore, the wheel can be used for nanoparticle transportation by using appropriate attachment chemistry. Loading and unloading mechanisms for cargos on the wheel can be designed using strand displacement as described in [138].

Another extension to polymerase based nanotransportation device is to make it programmable in the sense that it has capability of making decisions on choosing a path from amongst multiple paths. Equally important is to impart back and forth shuttling capabilities to the polymerase motor. Then a possible application of polymerase powered nanotransportation device can be in the construction of nanoshuttles. Arbitrary tracks analogous to the railway tracks can be laid out on nanostructures, and we might have multiple polymerase nanoshuttles working in tandem carrying out nanoscale transportation in a programmable and efficient manner.

Chapter 7

Conclusions

In this thesis, we focussed on two interlinked areas of DNA based nanotechnology, namely, DNA based self-assembly and DNA based nanorobotics. Self-assembly provides an excellent mechanism for bottom-up fabrication of nanoscale objects. Nanorobotical devices that can act in programmable manner on those nanoscale objects can lead to the development of extremely powerful applications, that can change the face of the technology.

Fault-tolerance in self-assembly is one of the most important challenges in self-assembly as errors in self-assembly are a major detrimental factor in the progress in nanocomputing and nanofabrication. A comprehensive theory of error-resilience is useful in design of fault-tolerant self-assemblies that are compact in nature. At the same time, the development of novel and stronger self-assembly models is necessary to realize the potential of self-assembly to a greater extent. Our time-dependent glue model provides an instance of a powerful self-assembly model capable of constructing complex structures using fewer building blocks and demonstrate phenomena, which are otherwise very difficult.

In the area of nanorobotics, a good modeling tool for nanorobotical devices is desperately needed. It can provide immensely useful foresight, while designing novel nanodevices, thus saving important research dollars and valuable time of scientists. A comprehensive framework for modeling DNA nanorobotical devices lays an ideal platform for creation of such a modeling tool. The novel designs of various protein-less, autonomous and programmable DNAzyme based devices provides new ideas for design of sophisticated DNA based nanodevices. The experimental demonstration of efficient fast moving polymerase based nanomotor open completely new frontiers in DNA based nanotechnology.

In effect, (1) by studying the self-assembly process analytically, insilico and experi-

mentally, and (2) by designing, analyzing and implementing novel nanorobotical systems, we have tried to address the challenges and the fundamental questions raised in the Chapter 1, namely:

- How to control errors in self-assembly ?
- How to construct complex nanoscale objects in simpler ways?
- How to transport nanoscale objects in programmable manner?

In this Chapter, we would like to discuss a concluding summary of our contributions in these aspects, and provide a roadmap for further advancements.

Fault Tolerance in Self-Assembly

The first contribution of this thesis is extensive study of compact error resilient schemes. We present a theoretical analysis of redundancy based compact error resilient tiling in two and three dimensions. First we presented a compact error correction schemes in two dimensional self-assembly that reduces the error from ϵ to ϵ^2 for arbitrary Boolean functions. Then we characterized the class of Boolean functions for which error reduction from ϵ to ϵ^3 is possible using redundancy based compact error resilient schemes. We also proved that error reduction from ϵ to ϵ^4 is impossible using redundancy based compact error resilient schemes. Next we examined three-dimensional self-assembly. First we presented a compact error resilient scheme that reduces error to ϵ^2 for arbitrary Boolean functions and ϵ^3 for a restricted class of input-sensitive Boolean functions. We also proved that error reduction to ϵ^4 can not be obtained for arbitrary Boolean functions using redundancy based compact error resilient schemes. We conjectured the following stronger results for three-dimensional assemblies that are currently open questions to be proved or disproved:

Conjecture: For arbitrary Boolean functions f_1, f_2, and f_3, there exists no redundancy

based compact error correction scheme that will reduce error from ϵ to ϵ^3 in three-dimensional self-assembly. □

Conjecture: For any functions f_1, f_2, and f_3 that are outside the restricted class of the functions defined in Theorem 6 there exists no redundancy based compact error correction scheme that will reduce error from ϵ to ϵ^3 in three-dimensional self-assembly. □

Conjecture: For any Boolean functions f_1, f_2, and f_3, , there exists no redundancy based compact error resilient scheme that can reduce error from ϵ to ϵ^4 in three-dimensional self-assembly. □

We have presented a three-dimensional extension to Winfree's self-healing tile set in two-dimensions[192]. It remains an open question if it is possible to design a compact self-healing tile set for two and three-dimensional self-assembly.

A Self-Assembly Model of Time Dependent Glue

The development of new and powerful self-assembly models can assist in realization of complex nanostructures and phenomena using building blocks designed with relative ease. We defined a model built on the basic framework of Tile Assembly Model [148], in which the glue strength between tiles depends upon the time they have been abutting each other. It can be implemented using strand displacement reactions on DNA tiles. Under this model, we demonstrated and analyzed catalysis and self-replication, and showed construction of a thin $k \times N$ rectangle using $O(\frac{\log N}{\log \log N})$ tiles for constant $k > 0$. The upper bound on assembling a thin rectangle was obtained by applying similar assembly strategy as in the multi-temperature model [9]. An interesting open question is whether the multi-temperature model can be simulated using our time-dependent model. It is also an open problem if under our model the lower bound of $\Omega(\frac{\log N}{\log \log N})$ for the tile complexity of an $N \times N$ square can be further improved.

Another interesting problem is to study the kinetics of the catalysis and self-replication analytically. Winfree's kinetic model [190] can be used to study them, but the challenge here is that the rate constant for the dissociation for a particular species varies with time because of changing glue strengths of its bonds. This makes the analytical study hard. However, these catalytic and self-replicating systems can be modeled as a continuous time markov chain, and studied using computer simulation to obtain empirical results.

Framework for Modeling DNA based Nanorobotical Devices

The design process of DNA based nanomechanical devices can be made considerably more efficient and robust with the help of simulators that can model such systems accurately prior to their experimental implementation. We presented a comprehensive framework for building a software tool for simulating a DNA based molecular system, and not the actual software tool.

We believe that the methods presented in Chapter 4 make a good framework for designing the simulator for DNA based molecular systems. We have described how to capture geometric constraints of the molecules with the polymer theory and MC simulation. The preliminary results in the chapter support the feasibility of the approach. We also described the approximations and limitations in this framework and the ways of improving them.

It is important to note that, as a framework, the physical simulation component and event simulation component can be decoupled as each component is improved individually.

A further extension to our framework would be to consider more complicated interactions, i.e. the enzyme restriction event and the hairpin formation. Another extension is to incorporate sequence design capabilities to the system. We would like to design and optimize sequences based on the given nanostructure conformations. A conformational change in a nanodevice can be decomposed into units of local deformations to ease the sequence design.

DNAzyme based Autonomous Nanodevices

We have described the construction of various devices based on the DNAzymes in Chapter 5. We designed (1) DNAzyme FSA, a finite state automaton using DNAzyme, (2) DNAzyme router, a device for programmable routing on two-dimensional nanostructures, and (3) DNAzyme doctor, a medical related device that releases a drug RNA based on the underexpression and overexpression of other RNAs. We focused more towards novel designs of the nanodevices with sophisticated fuctionalities rather than the details of the laboratory implementation. However, it should be noted that among other implementational challenges, the construction of the desired bulge loops for input nanostructures needs further investigation.

DNAzymes evolve through invitro selection procedures, and these processes can be designed to generate DNAzymes that cut distinct sequences. In the DNAzyme FSA, the number of DNAzymes required is proportional to the number of transitions in the automata. For binary-coded inputs the number of transitions is proportional to number of states. However, the implementation of finite state machines that do not have a planar layout might be challenging. Thus, it is a question for further research if this scheme can be extended easily to the design of finite state automata, whose layout is non-planar. The molecular computer for logical control of RNA expression can be useful in medical field if it can be used inside a cell, and the programmable walkers can be a really useful tool in nanopartical transportation systems at nanoscale. In conclusion, the designs in this chapter might pave way for evolution of sophisticated nanodevices.

Nanomotor Powered by Polymerase

The nanomotor powered by polymerase is a novel concept. We have been aware of the role of polymerase in sustaining life cycles for quite some time, but to exploit its mechanical energy directly in transportation of nanomaterial has been demonstrated for the first

time. The exceptional strand displacement abilities of polymerase $\phi 29$ are the basis of our nanomotor. It is noteworthy that this nanomotor has intrinsic advantage in speed as compared to other known synthetic molecular motors due to the high speeds of polymerase. The rate has been estimated to be approximately equal to 2000 bases per minute which translates to 1400 nanometers per minute on a nanostructure.

It would be interesting to experiment this nanomotor on two dimensional structures. A major challenge in accomplishing that goal is the tendency of $\phi 29$ to destroy the track (by producing its complementary strand hybridized to it). It might result in the detachment of the track from the two-dimensional nanostructure on which it is laid out in a particular desired shape. Ideally we would like to preserve the track for multiple runs of the nanomotor. Another problem is the potential extension (due to the polymerase $\phi 29$) of other single strands that constitute the required platform nanostructure. This can threaten the destruction of underlying platform nanostructures, given the strand displacement abilities of $\phi 29$.

However, a possible remedy for such a situation is to extend each of those single strands with small overhangs that are not complementary to the strands they are hybridized with currently. A foolproof but costly solution is to insert ddNTPs at the ends of such single strands. As a byproduct of the series of experiments, we also discovered that the polymerase $\phi 29$ cannot extend the primer beyond a nick in the template. We also observed that $\phi 29$ shows less fidelity if present in higher concentrations.

In conclusion, this work attempts to address various issues in interconnected and mutually dependent areas of DNA based nanotechnology. The development in each of them is necessary for the overall growth of the field. We would consider this work successful, if we are able to provide motivation and direction towards evolution of a few new frontiers and success on existing frontiers. And then, all of us can move further to explore the untravelled road ahead in this exciting area of DNA based nanotechnology.

Bibliography

[1] http://mrsec.wisc.edu/edetc/selfassembly/.

[2] http://www.neb.com/nebecomm/products/faqproductm0269.asp.

[3] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[4] L. Adleman. Towards a mathematical theory of self-assembly. Technical Report 00-722, University of Southern California, 2000.

[5] L. Adleman, Q. Cheng, A. Goel, and M.D. Huang. Running time and program size for self-assembled squares. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 740–748. ACM Press, 2001.

[6] L. Adleman, Q. Cheng, A. Goel, M.D. Huang, D. Kempe, P.M. de Espans, and P.W.K. Rothemund. Combinatorial optimization problems in self-assembly. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 23–32. ACM Press, 2002.

[7] L. Adleman, Q. Cheng, A. Goel, M.D. Huang, and H. Wasserman. Linear self-assemblies: Equilibria, entropy, and convergence rate. In *Sixth International Conference on Difference Equations and Applications*, 2001.

[8] L. Adleman, J. Kari, L. Kari, and D. Reishus. On the decidability of self-assembly of infinite ribbons. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 530–537, 2002.

[9] G. Aggarwal, Q. Cheng, M. H. Goldwasser, M. Kao, P. M. de Espanes, and R. T. Schweller. Complexities for generalized models of self-assembly. *SIAM Journal of Computing*, 24:1493–1515, 2005.

[10] P. Alberti and J. L. Mergny. DNA duplex-quadruplex exchange as the basis for a nanomolecular machine. *Proc. Natl. Acad. Sci. USA*, 100:1569–1573, 2003.

[11] S. A. Allison and S. Mazur. Modeling the free solution electrophoretic mobility of short dna fragments. *Biopolymers.*, 46:359–373., 1998.

[12] S. A. Allison and J. A. McCammon. Multistep brownian dynamics: application to short wormlike chains. *Biopolymers*, 23:363–375, 1984.

[13] S. R. Aragon and R. Pecora. Dynamics of wormlike chains. *Macromolecules*, 18(10):1868–1875, 1985.

[14] R. G. C. Arridge. *An introduction to polymer mechanics*. 1985.

[15] G. A. Arteca, T. Edvinsson, and C. Elvingson. Compaction of grafted wormlike chains under variable confinement. *Phys. Chem. Chem. Phys.*, 3:3737–3741, 2001.

[16] A. Aviram and M. Ratner. *Molecular Electronics: Science and Technology*. New York Academy of Sciences, New York, 1998.

[17] J Baner, M Nilsson, M Mendel-Hartvig, and U Landegren. Signal amplification of padlock probes by rolling circle replication. *Nucleic Acids Research*, 26:5073–5078, 1998.

[18] J. Bath, S. J. Green, and A. J. Turberfield. A free-running DNA motor powered by a nicking enzyme. *Angew. Chem. Intl. Ed.*, 44:4358–4361, 2005.

[19] W. R. Bauer, R. A. Lund, and J. H. White. Twist and writhe of a dna loop containing intrinsic bends. *Proc Natl Acad Sci U S A*, 90:833–837, 1993.

[20] Y. Benenson, R. Adar, T. Paz-Elizur, Z. Livneh, and E. Shapiro. DNA molecule provides a computing machine with both data and fuel. *Proc. Natl. Acad. Sci. USA*, 100:2191–2196, 2003.

[21] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429:423–429, 2004.

[22] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414:430–434, 2001.

[23] R. Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66, 1966.

[24] A. J. Berman, S. Kamtekar, J. L. Goodman, J. M. Lzaro, M. de Vega, L. Blanco, M. Salas, and T. A. Steitz. Structures of phi29 dna polymerase complexed with substrate: the mechanism of translocation in b-family polymerases. *EMBO Journal*, 26:3494–3505, 2007.

[25] I. Biswas, A. Yamamoto, and P. Hsieh. Branch migration through dna sequence heterology. *J. Mol. Bio*, 1998.

[26] R. D. Blake and S. G. Delcourt. Loop energy in dna. *Biopolymers*, 26:2009–2026, 1987.

[27] J. S. Bois, S. Venkataraman, H. M. T. Choi, A. J. Spakowitz, Z. Wang, and N. A. Pierce. Topological constraints in nucleic acid hybridization kinetics. *Nucleic Acids Research*, 33(13):4090–4095, 2005.

[28] M. Bonaccio, A. Credali, and A. Peracchi. Kinetic and thermodynamic characterization of the rna-cleaving 8-17 deoxyribozyme. *Nucleic Acids Res*, 32:916 – 925, 2004.

[29] B. A. Bondarenko. *Generalized Pascal Triangles and Pyramids, Their Fractals, Graphs and Applications*. The Fibonacci Association, 1993. Translated from Russion and edited by R.C. Bollinger.

[30] J. Borlido, S. Pereira, R. Ferreira, N. Coelho, P. Duarte, and J. Pissarra. Simple and fast in situ hybridization. *Plant Molecular Biology Reporter*, 20:219–229, 2002.

[31] C. Bouchiat, M. D. Wang, J. Allemand, T. Strick, S. M. Block, and V. Croquette. Estimating the persistence length of a worm-like chain molecules from force-extension measurements. *Biophys. J.*, 76:409–413, January 1999.

[32] N. Bowden, A. Terfort, J. Carbeck, and G.M. Whitesides. Self-assembly of mesoscale objects into ordered two-dimensional arrays. *Science*, 276(11):233–235, 1997.

[33] R.F. Bruinsma, W.M. Gelbart, D. Reguera, J. Rudnick, and R. Zandi. Viral self-assembly as a thermodynamic process. *Phys. Rev. Lett.*, 90(24):248101, 2003 June 20.

[34] J. R. Buchi. Turing machines and entscheidungsproblem. *Mathematiche Annalen*, 148:201–213, 1962.

[35] L. A. Bumm, J. J. Arnold, L. F. Charles, T. D. Dunbar, D. L. Allara, and P. S. Weiss. Directed self-assembly to create molecular terraces with molecularly sharp boundaries in organic monolayers. *J. Am. Chem. Soc.*, 121:8017 –8021, 1999.

[36] C. Bustamante, J. F. Marko, E. D. Siggia, and S. Smith. Entropic elasticity of lambda-phage dna mechanics. *Science*, 265:1599, 1994.

[37] C. Bustamante, S. Smith, J. Liphardt, and D. Smith. Single-molecule studies of dna mechanics. *Current Opinion in Structural Biology*, 10:279–285, 2000.

[38] J. E. Butler and E. S. G. Shaqfeh. Brownian dynamics simulations of a flexible polymer chain which includes continuous resistance and multi-body hydrodynamic interaction. *Journal of Chemical Physics*, 122:14901, 2005.

[39] M. J. Cairns, A. King, and L. Sun. Optimisation of the 10-23 dnazyme-substrate pairing interactions enhanced rna cleavage activity at purine-cytosine target sites. *Nucleic acids research*, 31:2883–2889, 2003.

[40] G. A. Carri and M. Marucho. Statistical mechanics of worm-like polymers from a new generating function. *J. Chem. Phys.*, 121(12):6064–6077, 2004.

[41] N. Chelyapov, Y. Brun, M. Gopalkrishnan, D. Reishus, B. Shaw, and L. Adleman. DNA triangles and self-assembled hexagonal tilings. *J. Am. Chem. Soc.*, 126:13924–13925, 2004.

[42] H.L. Chen, Q. Cheng, A. Goel, M.D. Huang, and P.M. de Espanes. Invadable self-assembly: Combining robustness with efficiency. In *Proceedings of the 15th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 890–899, 2004.

[43] H.L. Chen and A. Goel. Error free self-assembly using error prone tiles. *Lecture Notes in Computer Science*, 3384:62–75, 2005.

[44] J. Chen, M. A. Reed, A. M. Rawlett, and J. M. Tour. Large on-off ratios and negative differential resistance in a molecular electronic device. *Science*, 286:1550–1552, 1999.

[45] Y. Chen and C. Mao. Putting a brake on an autonomous DNA nanomotor. *J. Am. Chem. Soc.*, 126:8626–8627, 2004.

[46] Y. Chen, M. Wang, and C. Mao. An autonomous DNA nanomotor powered by a DNA enzyme. *Angew. Chem. Int. Ed.*, 43:3554–3557, 2004.

[47] Q. Cheng and P.M. de Espanes. Resolving two open problems in the self-assembly of squares. Technical Report 03-793, University of Southern California, 2003.

[48] Q. Cheng, A. Goel, and P. Moisset. Optimal self-assembly of counters at temperature two. In *Proceedings of the first conference on Foundations of nanoscience: self-assembled architectures and devices*, 2004.

[49] T. D. Clark, R. Ferrigno, J. Tien, K. E. Paul, and G. M. Whitesides. Template-directed self-assembly of 10-microm-sized hexagonal plates. *J Am Chem Soc.*, 124(19):5419–26, 2002.

[50] S. Cocco, J. F. Marko, and R. Monasson. Theoretical models for single-molecule dna and rna experiments: from elasticity to unzipping. *C. R. Physique*, 3:569–584, 2002.

[51] M. Cook, P. W. K. Rothemund, and E. Winfree. Self-assembled circuit patterns. In *DNA Based Computers 9*, volume 2943 of *LNCS*, pages 91–107, 2004.

[52] C. Desruisseaux, D. Long, G. Drouin, and G. W. Slater. Electrophoresis of composite molecular objects. 1. relation between friction, charge and ionic strength in free solution. *Macromolecules*, 34:44–59, 2001.

[53] M. N. Dessinges, B. Maier, Y. Zhang, M. Peliti, D. Bensimon, and V. Croquette. Stretching single stranded dna, a model polyelectrolyte. *Phys. Rev. Lett.*, 89:248102, 2002.

[54] P. Dimitrakopoulos. Stress and configuration relaxation of an initially straight flexible polymer. *J. Fluid Mech.*, 513:265–286, 2004.

[55] R. M. Dirks, J. S. Bois, J. M. Schaeffer, E. Winfree, and N. A. Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM Rev*, 49 (1):65–88, 2007.

[56] P. S. Doyle and P. T. Underhill. Brownian dynamics simulations of polymers and soft matter. In S. Yip, editor, *Handbook of Materials Modeling*, pages 2619–2630. Springer, 2005.

[57] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. 1968.

[58] L. Feng, S. H. Park, J. H. Reif, and H. Yan. A two-state DNA lattice switched by DNA nanoactuator. *Angew. Chem. Int. Ed.*, 42:4342–4346, 2003.

[59] M. Fixman and J. Kovac. Polymer conformation statistics iii: Modified gaussian models of the stiff chains. *J. Chem. Phys.*, 58:1564–1568, 1973.

[60] C. Flamm, W. Fontana, I. L. Hofacker, and P. Schuster. Rna folding at elementary step resolution. *RNA*, 6(3):325–38, 2000.

[61] J. B. Fournier. Wormlike chain or tense string? a question of resolution. *Continuum Mechanical Thermodynamics*, 14:241, 2002.

[62] M.D. Frank-Kamenetskii. Biophysics of dna molecule. *Phys. Rep.*, 288:13 – 60, 1997.

[63] K. Fujibayashi and S. Murata. A method for error suppression for self-assembling DNA tiles. *Lecture Notes in Computer Science*, 3384:113–127, 2005.

[64] M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. 1977.

[65] M. Garzon, E. Drumwright, R. Deaton, and D. Renault. Virtual test tubes: a new methodology for computing. In *In Proceedings of 7th International Symposium on String Processing and Information Retrieval*, pages 116–121. IEEE Computer Society Pzzress, 2000.

[66] M. Garzon and C. Oehmen. Biomolecular computation in virtual test tubes. *Lecture Notes in Computer Science LNCS*, 2340:117–128, 2002.

[67] M. H. Garzon, D. R. Blain, and A. J. Neel. Virtual test tubes. *Natural Computing*, 3:461–477, December 2004.

[68] J. Gelles and R. Landick. Rna polymerase as a molecular motor. *Cell*, 93:13–16, 1998.

[69] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977.

[70] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115:1716–1733, 2001.

[71] A. Goel, R. D. Astumian, and D. Herschbach. Tuning and switching a dna polymerase motor with mechanical tension. *Proc Natl Acad Sci U S A*, 100:9699–9704, 2003.

[72] A. J. Hartemink and D. K. Gifford. Thermodynamics simulation of deoxyoligonucleotide hybridization for dna computation. In *3rd Annual DIMACS workshop on DNA-based computers*, pages 15–25, 1997.

[73] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[74] P. J. Heath, J. A. Gebe, S. A. Allison, and J. M. Schurr. Comparison of analytical theory with brownian dynamics simulations for small linear and circular dnas. *Macromolecules*, 29:3583, 1996.

[75] B. D. Hughes. *Random Walks and Random Environments, Vol. 1: Random Walks.* New York: Oxford University Press, 1995.

[76] J. S. Hur and E. S. G. Shaqfeh. Brownian dynamics simulations of single dna molecule in shear flow. *J. Rheol.*, 44(4):713–742, July-August 2000.

[77] N. Ichinose. Hybrisim: Dna hybridization simulator. http://www.genome.ist.i.kyoto-u.ac.jp/ ichinose/bio/hybrisim/.

[78] H. Isambert and E. D. Siggia. Modeling rna folding paths with pseudoknots: application to hepatitis delta virus ribozyme. *Proc Natl Acad Sci U S A.*, 97(12):6515–20, 2000.

[79] Li J., W. Zheng, A.H. Kwon, and Y Lu. In vitro selection and charcterization of a highly efficient zn(ii) dependent, rna-cleaving deoxyribozyme. *Nucleic Acids Res.*, 28:481–488, 2000.

[80] H. M. James and E. Guth. Theory of the elastic properties of rubber. *Journal of Chemical Physics*, 10:455–481, 1943.

[81] R. M. Jendrejack, J. J. Pablo, and M. D. Graham. Stochastic simulations of dna in flow: Dynamics and the effects of hydrodynamic interactions. *Journal of Chemical Physics*, 116(17):7752, 2002.

[82] N. Jonoska, S. A. Karl, and M. Saito. Three dimensional DNA structures in computing. *BioSystems*, 52:143–153, 1999.

[83] J. SantaLucia Jr. A unified view of polymer, dumbbell and oligonucleotide dna nearest-neighbor thermodynamics. *PNAS*, 95:1460–1465, 1998.

[84] S. Kamtekar, A. J. Berman, J. Wang, J. M. Lzaro, M. de Vega, L. Blanco, M. Salas, and T. A. Steitz. Insights into strand displacement and processivity from the crystal structure of the protein-primed dna polymerase of bacteriophage phi29. *Mol Cell.*, 16:609–18, 2004.

[85] S. Kamtekar, A. J. Berman, J. Wang, J. M Lzaro, M. de Vega, L. Blanco, M. Salas, and T. A. Steitz. The phi29 dna polymerase:protein-primer structure suggests a model for the initiation to elongation transition. *EMBO J.*, 2006.

[86] M. Kao and R. Schweller. Reduce complexity for tile self-assembly through temperature programming. In *Proceedings of 17th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 571–580. ACM Press, 2006.

[87] A. M. Kierzek. Stocks: Stochastic kinetic simulations of biochemical systems with gillespie algorithm. *Bioinformatics*, 18:470–481, 2002.

[88] E. Klavins. Directed self-assembly using graph grammars. In *Foundations of Nanoscience: Self Assembled Architectures and Devices*, Snowbird, UT, 2004.

[89] E. Klavins, R. Ghrist, and D. Lipsky. Graph grammars for self-assembling robotic systems. In *Proceedings of the International Conference on Robotics and Automation*, 2004.

[90] K. Klenin, H. Merlitz, and J. Langowski. A brownian dynamics program for the simulation of linear and circular dna and other wormlike chain polyelectrolytes. *Biophys J*, 74(2):780–788, February 1998.

[91] J. Kovac and C. Crabb. Modified gaussian model for rubber elasticity. 2. the wormlike chain. *Macromolecules*, 15(2):537, 1982.

[92] M. Kuhn and F. Grun. Relationships between elastic constants and stretching double refraction of highly elastic substances. *Kolloid-Z*, 101:294, 1942.

[93] S. Kutter. *Elasticity of polymers with internal topological constraints*. PhD thesis, August 2002.

[94] T. H. LaBean, H. Yan, J. Kopatsch, F. Liu, E. Winfree, J. H. Reif, and N. C. Seeman. The construction, analysis, ligation and self-assembly of DNA triple crossover complexes. *J. Am. Chem. Soc.*, 122:1848–1860, 2000.

[95] B. Ladoux, J. P. Quivy, P. S. Doyle, G. Almouzni, and J. L. Viovy. Direct imaging of single-molecules: from dynamics of a single dna chain to the study of complex dna-protein interactions. *Sci. Prog.*, 84:267, 2001.

[96] M.G. Lagoudakis and T.H. LaBean. 2-D DNA self-assembly for satisfiability. In *DNA Based Computers V*, volume 54 of *DIMACS*, pages 141–154. American Mathematical Society, 2000.

[97] J. Langowski. Polymer chain models of dna and chromatin. *The European Physical Journal E,* 19:241–249, March 2006.

[98] R. G. Larson, H. Hu, D. E. Smith, and S. Chu. Brownian dynamics simulation of a dna molecule in an extensional flow field. *J. Rheol.,* 43(2):267–304, March-April 1999.

[99] R. G. Larson, T. Perkins, D. Smith, and S. Chu. Hydrodynamics of a dna molecule in a flow field. *Phys. Rev. E.,* 55:1794–1797, 1997.

[100] R. G. Larson, T. T. Perkins, D. E. Smith, and S. Chu. Brownian dynamics simulations of a dna molecule in an extensional flow field. *J. Rheol.,* 43:267, 1999.

[101] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1981.

[102] J. Li and W. Tan. A single DNA molecule nanomotor. *Nano Lett.,* 2:315–318, 2002.

[103] D. Liu and S. Balasubramanian. A proton fuelled DNA nanomachine. *Angew. Chem. Int. Ed.,* 42:5734–5736, 2003.

[104] D. Liu, M. Wang, Z. Deng, R. Walulu, and C. Mao. Tensegrity: Construction of rigid DNA triangles with flexible four-arm dna junctions. *J. Am. Chem. Soc.,* 126:2324–2325, 2004.

[105] Q. Liu, L. Wang, A. G. Frutos, A. E. Condon, R. M. Corn, and L. M. Smith. DNA computing on surfaces. *Nature*, 403:175–179, 2000.

[106] B. Maier, D. Bensimon, and V. Croquette. Replication by a single dna polymerase of a stretched single-stranded dna. *Proc. Natl. Acad. Sci. U.S.A.,* 97(22):12002–7, October 2000.

[107] A. Malevanets and J. M. Yoemans. Dynamics of short polymer chains in solution. *Europhysics Letters*, 52(2):231, 2000.

[108] C. Mao, T. H. LaBean, J. H. Reif, and N. C. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature,* 407:493–496, 2000.

[109] C. Mao, W. Sun, and N. C. Seeman. Designed two-dimensional DNA holliday junction arrays visualized by atomic force microscopy. *J. Am. Chem. Soc.,* 121:5437–5443, 1999.

[110] C. Mao, W. Sun, Z. Shen, and N. C. Seeman. A DNA nanomechanical device based on the B-Z transition. *Nature,* 397:144–146, 1999.

[111] J. F. Marko and E. D. Siggia. Bending and twisting elasticity of dna. *Macromolecules*, 27:981, 1994.

[112] J. F. Marko and E. D. Siggia. Stretching dna. *Macromolecules*, 28:8759, 1995.

[113] B. R. Martin, D. C. Furnange, T. N. Jackson, T. E. Mallouk, and T. S. Mayer. Self-alignment of patterned wafers using capillary forces at a water-air interface. *Advanced Functional Materials*, 11:381–386, 2001.

[114] D. Matsuda and M. Yamamura. Cascading whiplash pcr with a nicking enzyme. *Lecture Notes In Computer Science*, 2568:38 – 46, 2002.

[115] R. J. Meagher, J. Won, L. C. McCormick, S. Nedelcu, M. M. Bertrand, J. L. Bertarm, G. Drouin, A. E. Barron, and G. W. Slaters. End-labeled free-solution electrophoresis of dna. *Electrophoresis*, 26:331–350, 2005.

[116] J. Mercier and G. W. Slater. Solid phase dna amplification: a brownian dynamics study of crowding effects. *Biophysical Journal*, 89:32–42, July 2005.

[117] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[118] M. C. Murphy, I. Rasnik, W. Cheng, T. M. Lohman, and T. Ha. Probing single-stranded dna conformation flexibility using fluorescence spectroscopy. *Biophysical Journal*, 86:2530–2537, April 2004.

[119] C. M. Niemeyer and M. Adler. Nanomechanical devices based on DNA. *Angew. Chem. Int. Edit.*, 41:3779–3783, 2002.

[120] M Nilsson, H. Malmgren, M. Samiotaki, M. Kwiatkowski, B. P. Chowdhary, and U. Landegren. Padlock probes: circularizing oligonucleotides for localized dna detection. *Science*, 265:2085–2088, 1994.

[121] A. Nishikawa, M. Hagiya, and M. Yamamura. Virtual dna simulator and protocol design by ga. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'99*, volume 2, pages 1810–1816, 1999.

[122] A. Nishikawa, M. Yamamura, and M. Hagiya. Dna computation simulator based on abstract bases. *Soft Computing*, 5:25–38, 2001.

[123] T. Odijk. Stiff chains and filaments under tension. *Macromolecule*, 28:7016–7018, 1995.

[124] I.G. Panyutin and P. Hsieh. The kinetics of spontaneous dna branch migration. *Proc Natl Acad Sci U S A.*, 91(6):2021–5, 1994 Mar 15.

[125] S. H. Park, C. Pistol, S. J. Ahn, J. H. Reif, A. R. Lebeck, C. Dwyer, and T. H. LaBean. Finite-size, fully addressable dna tile lattices formed by hierarchical assembly procedures. *Angew. Chem. Int. Ed.*, 45:735–739, 2006.

[126] P. J. Paukstelis, J. Nowakowski, J. J. Birktoft, and N. C. Seeman. Crystal structure of a continuous three-dimensional DNA lattice. *Chemistry and Biology*, 11:1119–1126, 2004.

[127] J. S. Pedersen, M. Laso, and P. Schurtenberger. Monte carlo study of excluded volume effects in wormlike micelles and semiflexible polymers. *Phys Rev E.*, 54(6):5917–5920, December 1996.

[128] M. C. Petty, M. R. Bryce, and D. Bloor. *An introduction to Molecular Electronics*. Oxford University Press, New York, 1995.

[129] N. Peyret, P. A. Seneviratne, H. T. Allawi, and J. Santalucia. Nearest-neighbor thermodynamics and nmr of dna sequences with internal aa,cc,gg and tt mismatches. *Biochemistry*, 38:3468, 1999.

[130] C. Rao and A. Arkin. Stochastic chemical kinetics and the quasi-steady-state assumption: application to the gillespie algorithm,. *J. of Chem. Phys.*, 118:4999–5010, 2003.

[131] M. A. Reed, C. Zhou, C. J. Muller, T. P. Burgin, and J. M. Tour. Conductance of a molecular junction. *Science*, 278:252– 254, 1997.

[132] J. H. Reif. Local parallel biomolecular computation. In H. Rubin and D.H. Wood, editors, *DNA-Based Computers 3*, volume 48 of *DIMACS*, pages 217–254. American Mathematical Society, 1999.

[133] J. H. Reif. The design of autonomous DNA nanomechanical devices: Walking and rolling DNA. *The 8th International Meeting on DNA Based Computers (DNA 8)*, 2002.

[134] J. H. Reif. The emergence of the discipline of biomolecular computation. *Biomolecular Computing, New Generation Computing*, 20(3):217–236, 2002.

[135] J. H. Reif. The design of autonomous DNA nanomechanical devices: Walking and rolling DNA. *Lecture Notes in Computer Science*, 2568:22–37, 2003. Published in Natural Computing, DNA8 special issue, Vol. 2, p 439-461, (2003).

[136] J. H. Reif, S. Sahu, and P. Yin. Complexity of graph self-assembly in accretive systems and self-destructible systems. *Lecture Notes in Computer Science*, pages 257–274, 2005.

[137] J. H. Reif, S. Sahu, and P. Yin. Compact error-resilient computational dna tilings. *Nanotechnology: Science and Computation*, pages 79–103, 2006.

[138] John H. Reif and Sudheer Sahu. Autonomous programmable dna nanorobotic devices using dnazymes. Technical Report CS-2007-06, Duke University, Computer Science Department, 2007.

[139] P. Revesz. *Random walk in random and non-random environments*. World Scientific Pub Co., 1990.

[140] M. Rief, H. Clausen-Schaumann, and H. E. Gaub. Sequence-dependent mechanics of single dna molecules. *Nature Structural Biology*, 6:346 – 349, 1999.

[141] R.M. Robinson. Undecidability and non periodicity of tilings of the plane. *Inventiones Math*, 12:177–209, 1971.

[142] I. Rodrguez, J. M. Lzaro, L. Blanco, S. Kamtekar, A. J. Berman, J. Wang, T. A. Steitz, M. Salas, and M. de Vega. A specific subdomain in phi29 dna polymerase confers both processivity and strand-displacement capacity. *Proc Natl Acad Sci U S A.*, 102:6407–12.

[143] John A. Rose, Russell J. Deaton, Masami Hagiya, and Akira Suyama. Pna-mediated whiplash pcr. *Lecture Notes In Computer Science*, 2340:104 – 116, 2001.

[144] P. W. K. Rothemund. Folding dna to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.

[145] P. W. K. Rothemund, N. Papadakis, and E. Winfree. Algorithmic self-assembly of DNA sierpinski triangles. *PLoS Biology 2 (12)*, 2:e424, 2004.

[146] P.W.K. Rothemund. Using lateral capillary forces to compute by self-assembly. *Proc. Natl. Acad. Sci. USA*, 97(3):984–989, 2000.

[147] P.W.K. Rothemund. *Theory and Experiments in Algorithmic Self-Assembly*. PhD thesis, University of Southern California, 2001.

[148] P.W.K. Rothemund and E. Winfree. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 459–468. ACM Press, 2000.

[149] P. Sa-Ardyen, N. Jonoska, and N. C. Seeman. Self-assembling DNA graphs. *Natural Computing*, 2:427–438, 2003.

[150] M. Sales-Pardo, R. Guimera, A. A. Moreira, J. Widom, and L. A. Amaral. Mesoscopic modeling for nucleic acid chain dynamics. *Phys Rev E Stat Nonlin Soft Matter Phys.*, 71:051902, 2005.

[151] J. Santalucia and D. Hicks. The thermodynamics of dna structural motifs. *Annu. Rev. Biophys. Biomol. Struct.*, 33:415, 2004.

[152] S. W. Santoro and G. F. Joyce. Mechanism and utility of an rna-cleaving dna enzyme. *Biochemistry*, 37:13330–13342, 1998.

[153] S. Schroder, M. Hain, and K. Sterflinger. Colorimetric in situ hybridization (cish) with digoxigenin-labeled oligonucleotide probes in autofluorescent hybhomycetes. *Internatl. Microbiol.*, 3:183–186, 2000.

[154] R. Schulman, S. Lee, N. Papadakis, and E. Winfree. One dimensional boundaries for DNA tile self-assembly. In *DNA Based Computers 9*, volume 2943 of *LNCS*, pages 108–125, 2004.

[155] R. Schulman and E. Winfree. Programmable control of nucleation for algorithmic self-assembly. *LNCS*, 3384:319–328, 2005.

[156] R. Schulman and E. Winfree. Self-replication and evolution of DNA crystals. In *The 13th European Conference on Artificial Life (ECAL)*, 2005.

[157] N. C. Seeman. *De novo* design of sequences for nucleic acid structural engineering. *J. Biomol. Struct. Dyn.*, 8:573–581, 1990.

[158] N. C. Seeman. DNA in a material world. *Nature*, 421:427–431, 2003.

[159] R. Sha, R. Liu, D. P. Millar, and N. C. Seeman. Atomic force microscopy of parallel DNA branched junction arrays. *Chemistry and Biology*, 7:743–751, 2000.

[160] W. B. Sherman and N. C. Seeman. A precisely controlled DNA biped walking device. *Nano Lett.*, 4:1203–1207, 2004.

[161] J. S. Shin and N. A. Pierce. A synthetic DNA walker for molecular transport. *J. Am. Chem. Soc.*, 126:10834–10835, 2004.

[162] F. C. Simmel and B. Yurke. Using DNA to construct and power a nanoactuator. *Phys. Rev. E*, 63:041913, 2001.

[163] F. C. Simmel and B. Yurke. A DNA-based molecular device switchable between three distinct mechanical states. *Appl. Phys. Lett.*, 80:883–885, 2002.

[164] N Simon, N LeBot, D Marie, F Partensky, and D Vaulot. Fluorescent in situ hybridization with rrna-targeted oligonucleotide probes to identify small phytoplankton by flow cytometry. *Appl. Environ. Microbiol.*, 61:2506–2513, 1995.

[165] S. B. Smith, Y. Cui, and C. Bustamante. Overstretching b-dna: the elastic response of individual double-stranded and single-stranded dna molecules. *Science*, 271:795–799, Feb 1996.

[166] S. B. Smith, L. Finzi, and B. Bustamante. Direct mechanical measurements of the elasticity of single dna molecules by using magnetic beads. *Science*, 258:1122, 1992.

[167] D. Soloveichik and E. Winfree. Complexity of compact proofreading for self-assembled patterns. *LNCS*, 3892:305–324, 2006.

[168] D. Soloveichik and E. Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36:1544–1569, 2007.

[169] M. Somasi, B. Khomami, N. J. Woo, J. S. Hur, and E. S. G. Shaqfeh. Brownian dynamics simulations of bead-rod and bead-spring chains: numerical algorithms and coarse-graining issues. *J. Non-Newtonian Fluid Mech.*, 108:227–255, 2002.

[170] A. S. Spirin. Rna polymerase as a molecular machine. *Molecular Biology*, 36:153–159, 2002.

[171] E. Stellwagen and N. C. Stellwagen. Determining the electrophoretic mobility and translational diffusion coefficients of dna molecules in free solution. *Electrophoresis*, 23(16):2794–2803, 2002.

[172] C. Storm and P. C. Nelson. Theory of high-force dna stretching and overstretching. *Physical Review E.*, 67:051906, 2003.

[173] P. Thomen, P. J. Lopez, and F. Heslot. Unravelling the mechanism of rna-polymerase forward motion by using mechanical force. *Physical Review Letters*, 94, 2005.

[174] B. J. Thompson, M. N. Camien, and R.C.Warner. Kinetics of branch migration in double-stranded dna. *Proc Natl Acad Sci U S A*, 73(7):2299–303, 1976 Jul.

[175] Y. Tian, Y. He, Y. Chen, P. Yin, and C. Mao. Molecular devices - a DNAzyme that walks processively and autonomously along a one-dimensional track. *Angew. Chem. Intl. Ed.*, 44:4355–4358, 2005.

[176] Y. Tian and C. Mao. Molecular gears: A pair of DNA circles continously rolls against each other. *J. Am. Chem. Soc.*, 126:11410–11411, 2004.

[177] I. Tinoco and C. Bustamante. The effect of force on thermodynamics and kinetics of single molecule reactions. *Biophys Chem.*, 101-102(1):513–33, 2002.

[178] A. J. Turberfield, J. C. Mitchell, B. Yurke, Jr. A. P. Mills, M. I. Blakey, and F. C. Simmel. DNA fuel for free-running nanomachines. *Phys. Rev. Lett.*, 90:118102, 2003.

[179] A. J. Turberfield, B. Yurke, and Jr. A. P. Mills. DNA hybridization catalysts and molecular tweezers. *DNA5*, 2000.

[180] T. E. Turner, S. Schnell, and K. Burrage. Stochastic approaches for modelling in vivo reactions. *Computational Biology and Chemistry*, 2004.

[181] A. V. Vologodskii. Monte carlo simulation of dna topological properties. In *Topology in Molecular Biology*, Biological and Medical Physics, Biomedical Engineering, pages 23–41. Springer Berlin Heidelberg, 2006.

[182] A.F. Voter. *Introduction to kinetic monte carlo method*. Springer, NATO publishing unit, 2005.

[183] H. Wang. Proving theorems by pattern recognition ii. *Bell Systems Technical Journal*, 40:1–41, 1961.

[184] M. D. Wang, M. J. Schnitzer, H. Yin, R. Landick, J. Gelles, and S. M. Block. Force and velocity measured for single molecules of rna polymerase. *Science*, 1998.

[185] S. M. Waybright, C. P. Singleton, J. M. Tour, C. J. Murphy, and U. H. F. Bunz. Synthesis and self-assembly of an oligonucleotide-modified cyclobutadiene complex. *Organometallics*, 19:368–370, 2000.

[186] J. G. Wetmur and N. Davidson. Kinetics of renaturation of dna. *J. Mol. Biol.*, 31:349–370, 1968.

[187] G. M. Whitesides and B. Grzybowski. Self-assembly at all scales. *Science*, 295:2418 – 242, 2002.

[188] E. Winfree. Complexity of restricted and unrestricted models of molecular computation. In R. J. Lipton and E.B. Baum, editors, *DNA Based Computers*, volume 27 of *DIMACS*, pages 187–198. American Mathematical Society, 1995.

[189] E. Winfree. Complexity of restricted and unrestricted models of molecular computation. In R. J. Lipton and E.B. Baum, editors, *DNA Based Computers 1*, volume 27 of *DIMACS*, pages 187–198. American Mathematical Society, 1996.

[190] E. Winfree. Simulation of computing by self-assembly. Technical Report 1998.22, Caltech, 1998.

[191] E. Winfree. Whiplash pcr for $o(1)$ computing. Technical Report 1998.23, Caltech, 1998.

[192] E. Winfree. Self-healing tile sets. *Nanotechnology: Science and Computation*, pages 55–78, 2006.

[193] E. Winfree and R. Bekbolatov. Proofreading tile sets: Error correction for algorithmic self-assembly. In *DNA Based Computers 9*, volume 2943 of *LNCS*, pages 126–144, 2004.

[194] E. Winfree, T. Eng, and G. Rozenberg. String tile models for DNA computing by self-assembly. In *DNA Based Computers 6*, pages 63–88, 2000.

[195] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–544, 1998.

[196] E. Winfree, X. Yang, and N. C. Seeman. Universal computation via self-assembly of DNA: Some theory and experiments. In L.F. Landweber and E.B. Baum, editors, *DNA Based Computers II*, volume 44 of *DIMACS*, pages 191–213. American Mathematical Society, 1999.

[197] M. T. Wolfinger, W. A. Svrcek-Seiler, C. Flamm, I. L. Hofacker, and P. F. Stadler. Exact folding dynamics of rna secondary structures. *J.Phys.A: Math.Gen.*, 37:4731–4741, 2004.

[198] X. Xiong, Y. Hanein, J. Fang, Y. Wang, W. Wang, D. Schwartz, and K. Bohringer. Controlled multibatch self-assembly of microdevices. *Journal Of Microelectromechanical Systems*, 12:117–127, 2003.

[199] H. Yamakawa and T. Yoshizaki. Dynamics of helical wormlike chains. i. dynamic model and diffusion equation. *Journal of Chemical Physics*, 75(2):1016–1030, July 1981.

[200] H. Yan, L. Feng, T. H. LaBean, and J. H. Reif. Parallel molecular computation of pair-wise xor using DNA string tile. *J. Am. Chem. Soc.*, 125(47), 2003.

[201] H. Yan, T. H. LaBean, L. Feng, and J. H. Reif. Directed nucleation assembly of DNA tile complexes for barcode patterned DNA lattices. *Proc. Natl. Acad. Sci. USA*, 100(14):8103–8108, 2003.

[202] H. Yan, S. H. Park, G. Finkelstein, J. H. Reif, and T. H. LaBean. DNA-templated self-assembly of protein arrays and highly conductive nanowires. *Science*, 301(5641):1882–1884, 2003.

[203] H. Yan, X. Zhang, Z. Shen, and N. C. Seeman. A robust DNA mechanical device controlled by hybridization topology. *Nature*, 415:62–65, 2002.

[204] J. Yan and J. F. Marko. Localized single-stranded bubble mechanism for cyclization of short double helix dna. *Phys. Rev. Lett.*, 93(10):108108, September 2004.

[205] P. Yin, S. Sahu, A. J. Turberfield, and J. H. Reif. Design of autonomous DNA cellular automata. In *Proc. 11th International Meeting on DNA Computing*, pages 376–387, 2005.

[206] P. Yin, A. J. Turberfield, and J. H. Reif. Designs of autonomous unidirectional walking DNA devices. Technical Report CS-2004-01, Duke University, Computer Science Department, 2004.

[207] P. Yin, A. J. Turberfield, S. Sahu, and J. H. Reif. Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion. In *Proc. 10th International Meeting on DNA Computing*, pages 344–356, 2004.

[208] P. Yin, H. Yan, X. G. Daniell, A. J. Turberfield, and J. H. Reif. A unidirectional DNA walker moving autonomously along a linear track. *Angew. Chem. Int. Ed.*, 43:4906–4911, 2004.

[209] B. Yurke, A.P. Mills, and A.J. Turberfield. A molecular machine made of and powdered by DNA. *Biophysics*, 78:2629, 2000.

[210] B. Yurke, A.J. Turberfield, Jr. A.P. Mills, F.C. Simmel, and J.L. Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, 406:605–608, 2000.

[211] Y. Zhang, H. Zhou, and Z. Ou-Yang. Stretching single-stranded dna: Interplay of electrostatic, base-pairing, and base-pair stacking interactions. *Biophys J.*, 81:1133–1143, August 2001.

[212] C. Zhou. *Atomic and Molecular wires*. PhD thesis, Yale University, 1999.

Biography

Sudheer Sahu was born on August 27th, 1980 in Gwalior, India. After recieving a B. Tech in Computer Science and Engineering from Indian Institute of Technology, Delhi in 2002, Sudheer joined the Ph.D. program in the Department of Computer Science at Duke University. His Ph.D. thesis explores the area of DNA based nanotechnology, with focus on self-assembly and nanorobotics. His research interests also broadly span algorithms, complexity theory, stochastic modeling, and graph theory.