DNA-based String Rewrite Computational Systems

by

Julia Khodor

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2002

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
February 7, 2002

Certified by ....
Professor David K. Gifford
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .... [illegible] ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

DNA-based String Rewrite Computational Systems

by

Julia Khodor

Submitted to the Department of Electrical Engineering and Computer Science
on February 7, 2002, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

We describe a DNA computing system called programmed mutagenesis. prove that it is universal, and present experimental results from a prototype computation. DNA is a material with important characteristics, such as possessing all the information necessary for self-reproduction in the presence of appropriate enzymes and components, simple natural evolution mechanism, and miniature scale, all of which make it an attractive substrate for computation. For computer science, using single DNA molecules to represent the state of a computation holds the promise of a new paradigm of composable molecular computing. For biology, the demonstration that DNA sequences could guide their own evolution under computational rules may have implications as we begin to unravel the mysteries of genome encoding.

Programmed mutagenesis is a DNA computing system that uses cycles of DNA annealing, ligation, and polymerization to implement programmatic rewriting of DNA sequences. We report that programmed mutagenesis is theoretically universal by showing how Minsky's 4-symbol 7-state Universal Turing Machine can be implemented using a programmed mutagenesis system. Each step of the Universal Turing Machine is implemented by four cycles of programmed mutagenesis, and progress is guaranteed by the use of alternate sense strands for each rewriting cycle.

We constructed a unary counter, an example programmed mutagenesis system, and operated it through three cycles of mutagenesis to gather efficiency data. We showed that the counter operates with increasing efficiency, but decreasing overall yield. The measured efficiency of an in vitro programmed mutagenesis system suggests that the segregation of the products of DNA replication into separate compartments would be an efficient way to implement molecular computation. Naturally occurring phenomena such as gene conversion events and RNA editing processes are also discussed as possible manifestations of programmed mutagenesis-like systems.

Thesis Supervisor: Professor David K. Gifford
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

To my parents, Inna and Leonid Khodor, for making the trip. Every time.

To my husband, Mark Beloborodov, for surviving the experience without complaining (too much).

To our work in progress, our baby, whom we have been waiting to meet for a very long time, for setting an immovable deadline for the completion of this project, and making getting there seem less stressful. And for not kicking through the defense.

The author would like to thank Professor David K. Gifford for his support and guidance through the duration of this research project and for his help in editing this document.

The author would like to thank Professor Ronald R. Breaker for his helpful comments on the manuscript and for making the trip for the defense, in spite of Amtrak.

The author would like to thank Dr. Tom Knight and Professor Albert Meyer for helpful discussions and comments on the manuscript.

The author would like to thank Professors Maurice Fox and Leonard Lerman for the use of their laboratory space and for helpful advice.

Contents

List of Figures

List of Tables

Chapter 1

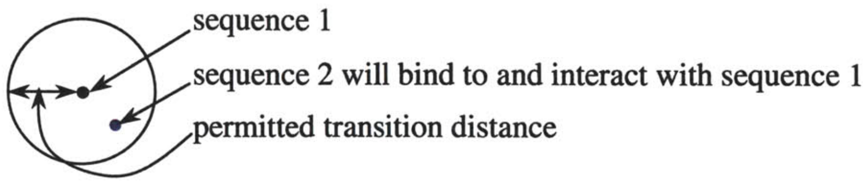
Introduction

DNA Computing has attracted interest since Adleman's original paper [2] because of its potential for high performance parallel computation. Adleman [3], and others have proposed that the intrinsic power of processing large numbers (10^8) of molecules in parallel may permit DNA computers to solve previously intractable problems. There have been a number of models proposed for DNA-based computing since Adleman's original publication. We will discuss in detail five main categories of these models after we introduce two important notions.

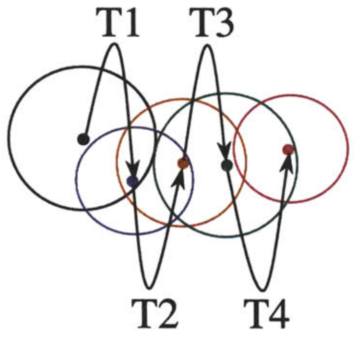
The first is the notion of transition space. A *transition space* of a given oligonucleotide sequence s_1 is the section of the sequence space around s_1 such that any sequence s_i within the transition space of s_1 can bind to and interact with s_1, and no sequence s_j outside of the transition space of s_1 can bind to and interact with s_1. Figure 1-1A illustrates the definition of the transition space. We first introduced transition space in 1998, under the name Hamming space [28].

Accordingly, as illustrated in Figure 1-1B and C, allowed transitions are those that rewrite a string s_i to a string s_j iff s_j is in the transition space of s_i.

All DNA computing systems rely on the notion of transition space. Some, like Adleman's original system [2], use the notion implicitly. The sequences in such a system are chosen such that the DNA oligonucleotides will not interact. Thus, the sequences are chosen such that no sequence s_i is in the transition space of any other sequence s_j. However, the design challenge for such systems is most often formulated a

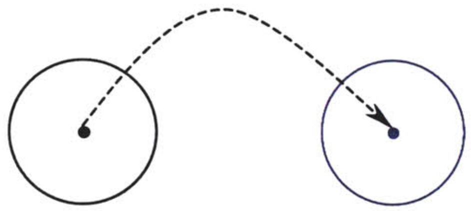

sequence 1
sequence 2 will bind to and interact with sequence 1
permitted transition distance

b



Transitions in sequence space

c



Disallowed transition in sequence space

Figure 1-1: Schematic representation of transition rewrite systems.
Part A shows transition space of the sequence 1, including sequence 2 in that transition space, which will bind to and interact with sequence 1.
Part B illustrates allowed transitions in sequence space. T1 through T4 are allowed because they accomplish transitions between sequences which each lie in the transition space of the sequence they rewrite.
Part C illustrates a disallowed transition in sequence space. Second sequence does not lie in the transition space of the first sequence in the figure.

simply as "choose sequences that do not interact." It could be argued that better understanding of the underlying principle of the transition space would help researchers in designing such systems.

Programmed mutagenesis systems use the notion of transition space explicitly. In programmed mutagenesis sequences are designed such that if a sequence s_1 is supposed to interact with a sequence s_2, but not a sequence s_3, then s_2, but not s_3 is in the transition space of s_1. Thus, programmed mutagenesis systems can be referred to as transition rewrite systems.

The second notion we introduce is the notion of a composable system. We call a computing system *composable* when a first computation results in a single molecule that can be used directly by a second computation as input without modification. The notion of a composable system is important, because invariants maintained in such a system would generally make its behavior more predictable since the format of the molecules in the system would be maintained throughout computation. In addition, such a system would cut down on the number of experimental manipulations required between steps of computation. Programmed mutagenesis systems are composable because both the input and output from a computation can be represented as a single strand of DNA. Programmed mutagenesis is the first composable system to be proven universal. A universal computational system is a system capable of performing any deterministic computation, that is a computation that given the same input always produces the same output.

Composable systems are an important class of DNA computing systems because it can be argued that composable systems are more likely to be autonomous, i.e. systems that can operate without human intervention during computation. Autonomous systems are desirable because any intervention increases the potential for experimental error. The source of such errors may be both human error and inherent error rate of the intervening processes. Programmed mutagenesis is an autonomous system, since it proceeds without human intervention between the cycles of computation. An additional interesting property of programmed mutagenesis is that all the machinery necessary to implement it is present in a cell.

In the remainder of the chapter, we will discuss the variety of approaches to DNA computing that have been proposed. These can be subdivided into four main categories.

1.1 Generate-and-search approaches

The generate-and-search approach, also referred to as combinatorial search approach was introduced by Adleman in 1994 [2]. This approach relies on generating a set of witness DNA molecules. Each witness molecule encodes a single solution to the problem being solved. Filters are applied to the set of witness molecules to select only those molecules that satisfy a given set of constraints, and are, therefore, solution molecules. Witness molecules are self-assembled out of the input oligonucleotides that represent various pieces of the problem, such as vertices and edges in a graph. This was the first experimental demonstration of any computational system based on DNA, but this approach is not universal. It is also not composable, since the input to a computation is a set of DNA oligonucleotides, and the output is a single molecule, assembled out of these oligonucleotides. Generate-and-search approach is not universal [71]. A related method, called the sticker-based model, which uses sequence-specific separation by hybridization as a central mechanism, is universal [51].

Combinatorial search relies on the idea that the immense storage density of DNA allows the use of brute-force search methods to solve NP-complete problems. For example, Lipton [39] suggests using combinatorial search approach to break the DES encryption code.

This approach has a number of experimental drawbacks [26, 29], although recent advances have been made in removing the need for human intervention between every step of the algorithm. A small instance of the satisfiability problem was solved on a gel-based computer [9], with separations performed using probes covalently bound to polyacrylamide gel. In this experiment, DNA was retained within a single gel and moved via electrophoresis. This is an important step towards autonomous combinatorial search algorithms. However, some of the problems associated with the separation procedure, as well as the problem of the loss of material as the computation proceeds undoubtedly persist in this implementation.

Most proposed implementations for the generate-and-search systems focus on molecules in solution. However, much work has been done to lay the foundation of surface-based computing. This approach involves the manipulation of DNA strands that have been immobilized on a surface [56]. Much work has been done in establishing the basic characteristics of such systems [41], as well as in carefully establishing the experimental criteria for successful word designs in such systems [18, 40]. Theoretical problem of word design arising from this approach, but extended to other areas of DNA computing, was also investigated [43]. It is interesting to note that parameter d– distance in mismatches between the code words used in the above work is remarkably similar to the simplified idea of transition space. Several algorithms to solve particular problems in NP have been proposed within surface-based framework [16, 46]. Morimoto and colleagues [46] also conducted some experiments to investigate the viability of their algorithm.

Surface-based chemistry has a number of advantages over solution-based chemistry, such as simplified handling of samples and reduced rate of loss of material through the stages of the computation. However, the major drawback of surface-based chemistry is the reduced density of information possible in such a system as compared to solution-based systems.

1.2 Splicing Systems

Splicing systems were pioneered by Tom Head [23] in 1987. The basic premise of the splicing systems is that when two strands of DNA are cut, generating four fragments (x_1, x_2 and y_1 and y_2), the pieces are recombined strictly in a crosswise manner (resulting in strands x_1y_2 and y_1x_2). This is in contrast with the known biological reality, and no mechanism presently exists to enforce such a restriction.

When splicing systems have finite sets of axioms and rules that define splicing, the systems are limited to generating regular languages [47]. If both of these sets remain finite, the only way to increase the computational power of the system is to introduce certain control mechanisms. A control mechanism is a mechanism which allows for only certain splicing rules to be active under certain circumstances. Splicing systems utilizing a number of such control mechanisms have been shown to be universal [12]. However, the control systems proposed require researcher's intervention and are vulnerable to experimental error. Furthermore, these mechanisms might prove difficult to implement in a laboratory. The splicing systems are composable, since the operations in the system simply recombine the cut pieces of the molecules in a crosswise manner.

The only experimental confirmation of the splicing systems was a small-scale experiment using two restriction enzymes and a ligase [36]. The system investigated generated the splicing language predicted by the corresponding dry splicing system. However, the extent to which this particular experimental approach can be generalized and remain autonomous is limited by the number of restriction enzymes which can be made to work in a common buffer. Any buffer used in such a system also needs to accommodate the ligase.

Several variants of the splicing systems have been introduced, including cutting/recombination (CR) schemes [17] and a number of systems arising from restricted use of the splicing operation [49]. CR-schemes differ from traditional splicing systems in that in these schemes cutting can occur independently from recombining the cut pieces. It is not clear, however, how to biologically enforce such a separation between cutting and recombination. In addition, these systems presuppose input filters for the test tubes for the redistribution of the contents of the test tubes after a period of cuttings and recombinations. Thus, CR systems are not autonomous. Restricted use of splicing operation, although not biologically inspired, provides for an interesting new look at the language theory, allowing for new characterizations for families in the Chomsky hierarchy and for closure properties in general.

P-systems, one of the possible control systems for splicing systems has been recently introduced by Gheorghe Paun [48] as a model for computation based on membrane structures. The basic premise is to use membranes as a filter for objects with specific properties when transferring them into an inner compartment, or out into surrounding compartment. Various P-systems using splicing or cutting and recombination rules have been shown to be universal, but it is not clear how to implement membrane control mechanisms biologically.

1.3 Systems Based on Self-assembly

The biological basis for computation via self-assembly of two-dimensional DNA structures was laid by the work of Seeman and colleagues, who constructed and analyzed double crossover molecules [37], and, later, triple crossover molecules [32]. Double crossover molecules are DNA structures containing two Holliday junctions connected by two double helical arms. Types of double crossover molecules are differentiated by the parallel or antiparallel orientations of their helix axes relative to each other, and by the number of double helical half-turns between the two crossovers, which could be even or odd.

A wide variety of language classes can be generated by self-assembly, depending on what kind of structures are used and in how many dimensions the assembly proceeds. Self-assembly of oligonucleotides into linear duplex is limited to generating regular languages, while branched DNA assembled into dendrimer structures is limited to generating context-free languages [73]. However, if hairpins are allowed, linear assembly can generate context-free languages [15]. Recently, it has been shown that if more complex linear tiles (termed string tiles) are assembled linearly, surprisingly sophisticated calculations can be performed. In fact, linear self-assembly of string tiles can generate the output languages of finite-visit Turing machines [69]. A number of algorithms for solving particular problems via self-assembly have also been proposed [34, 63].

Self-assembly of two-dimensional DNA structures was shown to be universal by Winfree [70] in 1995. Winfree also achieved experimental confirmation of the basic assembly operation of heterogeneous tiles [73, 72]. Winfree proposed to use ligation after self-assembly to produce a single reporter strand of DNA. This reporter strand would be used for reading the output of the self-assembly computation. It has also been proposed to use a single-stranded scaffold strand of DNA as an input molecule. The viability of assembling multiple tiles around such a scaffold was demonstrated [33]. Even if both an input scaffold and an output reporter strand are used, it is still not clear that a self-assembly system can be composable. This is because as the reporter strand weaves its way through the tiles assembled for computation, it changes directions, and thus, records the presence of some tiles in forward orientation, and some in reverse. Thus, it is not clear whether a reporter strand can then be used as a scaffold strand for a new computation.

A challenge in the experimental implementation of this approach might be that larger computations, which include more different varieties of tiles, will require longer time intervals for each step of the assembly because self-assembly relies on the correct tile assembly followed by the correct computational structure assembly. Further, minimizing the error rate also requires longer time intervals per assembly step. Recently, the first experimental demonstration of computation by self-assembly [42] has been produced. This is a significant advancement in DNA computing because it demonstrates the first DNA-only universal computational system which requires only ligase to create an output molecule. It is possible that a composable implementation of a self-assembly systems may still be proposed. However, no currently proposed self-assembly system, including the one experimentally implemented to date, is composable. In addition, for larger systems with greater number of tiles there also exist so far unresolved concerns regarding the possibility that locally assembled substructures may interfere with the formation of global structures [50], [68].

1.4 String-Rewrite Systems

Programmed mutagenesis is an example of a string-rewrite system for DNA computing. DNA naturally lends itself to a string rewrite model because the sequence of bases in DNA can directly be used to encode a string. DNA strand replication provides the ability to copy as well as the opportunity to introduce sequence specific changes into a newly synthesized molecule.

Since there is no known way to reliably internally mutate an existing DNA sequence using DNA polymerase, all DNA-based string-rewrite systems must include rewrite rules to be incorporated into the newly synthesized DNA strand. Thus, the main challenge in implementing any DNA-based string-rewrite system is to ensure the specificity of rewrite rules. This specificity is dependent on a particular implementation, as well as on the general restrictions posed by thermodynamics of the DNA-DNA and DNA-enzyme interactions.

String-rewrite systems can be subdivided into those where the rewrites occur by extension at the $3'$ end of the molecule and those where the rewrites are internal. Hairpin systems are an example of the former class of systems. Programmed mutagenesis is an example of the latter class.

Autonomous string systems based on hairpin formation were introduced by Hagiya *et al.* [20]. In these systems each molecule works not only as a data carrier but also as a computing unit. The system has been termed "whiplash", because extensions of a molecule are achieved by a series of hairpin formations followed by $3'$ extension, followed by breaking of the old and formation of the new hairpin. Some interesting experimental systems based on this principle have been constructed [30], [52]. It was shown that, combined with parallel overlap assembly, whiplash PCR can solve NP-complete problems [53], but theoretical universality of the system has not been demonstrated. These systems are also not composable, since the starting molecule represents the collection of the transitions allowed in the system (rewrite rules) separated by "stopper" sequences, that is sequences which can not be copied because the reaction mix lacks a nucleotide necessary to copy the segment, followed by the starting state, while the result molecule contains, in addition to all the information present on the start molecule, the entire history of computation, as well as the final state. The format of the starting and final molecules is not the same, since the final molecule only has stopper sequences in the original rule part of the molecule. In addition, since all the rules needed for computation are present in the molecule, it is difficult to see how a final molecule could be used to initiate a new computation, since presumably all the rules present on the molecule have been exercised and have led to a halt state.

Programmed mutagenesis is an example of an internal string-rewrite system for DNA computing. All such systems are composable, because the format of a molecule does not change during computation. However, not all such systems are autonomous. Systems requiring separation are obviously not autonomous. Both Beaver [5] and Kari *et al.* [27] have also proposed to use systems based on internal string-rewriting. While both of these models were proven to be theoretically universal, both have drawbacks:

1. Beaver's model proposes to use the substitution operation, that is rules of the form xyz $\rightarrow$ xuz. This model, however, requires separations after each substitution step to guard against the possibility of a rule performing a mixture of substitution and deletion, such as rewriting the sequence xyyz on the template into the sequence xuz, or insertion, such as rewriting the sequence xz on the template into the sequence xuz. Since separation operations are required, this model is also vulnerable to experimental error.

2. Kari *et al.* [27] proposed to use insertion/deletion ("insdel") systems to implement universal computation. While the insdel systems represent a theoretically interesting computational model, new techniques will be required to implement them in practice. With current techniques there are two problems with the proposed rules for the insdel systems. First, there is no way to control the length of the deleted sequences in the insdel systems. The deletion rules in the insdel systems are of the form xzy $\rightarrow$ xy, which with the currently available techniques have to be implemented by an oligonucleotide encoding sequence xy. This oligonucleotide would bind any occurrence of the contexts x and y, and would, therefore, delete any sequence embedded between these contexts. Thus, if the template read xzyy, the rule xy would, with about equal probability, delete sequences z and zy.

   Second, there is no way to prevent insertion rules in the insdel systems from performing substitutions. For example, an insertion rule of the form xz $\rightarrow$ xyz, represented by an oligonucleotide encoding the sequence xyz, and given a template xzzzz would be approximately equally likely to perform an insertion (rewriting the template into xyzzzz), or any of the three possible substitutions (rewriting the template into xyzzz, xyzz, or xyz).

An interesting implementation of a read-once string rewrite system was recently introduced by Benenson *et al.* [6]. The system relies on restriction enzyme-based rewrite rules to execute transitions of a finite state machine. A finite state machine can only process its input once and in order. Thus, when constructing a molecular finite state machine, one does not need to be concerned with implementing internal rewrites. It is not clear how to extend the methodology used in [6] to create molecules with internal rewrite events needed to implement string rewrite systems capable of reading and rewriting the same area of the input molecule multiple times.

1.5 Other Paradigms in DNA Computing

There are several other paradigms under research in the area of DNA computing. These paradigms can not be easily compared to those discussed above. Therefore, we only briefly mention these areas of research here.

Yurke and colleagues are interested in building molecular nanomachines. They have recently constructed DNA tweezers [60] and DNA scissors [45]. Both of these structures are self-assembled, and their operation ("opening" and "closing") is controlled by additional strands of DNA.

Jonoska and colleagues are researching the possibility of solving computational problems with DNA molecules by physically constructing 3-dimensional graph structures [24] or thickened graphs [25]. While these ideas are intriguing, no experimental study of their feasibility has been undertaken as of yet.

Several groups have looked into the idea of computing with cells. Collins and colleagues are developing a theory and an experimental protocol for constructing artificial gene networks that can regulate temporal expression of multiple genes. As a first step, several toggle switches were constructed [19]. Elowitz and Leibler have constructed a synthetic network of transcriptional regulators operation of which results in periodic expression of a fluorescent protein [14]. Weiss and Knight used the Lux operon of *Vibrio fischeri* to engineer intracellular communication mechanisms between living cells [65]. Wakabayashi and Yamamura propose to use the pheromone response system of *Enterococcus faecalis* to construct a pheromone-dependent DNA transfer system [64].

The remainder of this thesis is organized as follows. In Chapter 2, we introduce the computational paradigm of programmed mutagenesis and discuss its formal model, as well as a unary counter– an example experimental system. In Chapter 3, we present the proof of universality of programmed mutagenesis by demonstrating a direct encoding of the smallest known Universal Turing machine in DNA under the formal model presented in Chapter 2. We also discuss how the methodology used to create this particular encoding can be used to create encodings for other special purpose and Universal Turing machines. In Chapter 4, we present experimental results of operating the unary counter and discusses these results in terms of the feasibility of using programmed mutagenesis techniques for various applications. Finally, in Chapter 5, we discuss the results presented in the thesis, as well as examine some naturally occurring phenomena and the computational themes apparent therein.

Chapter 2

Programmed Mutagenesis and Unary Counter

2.1 Overview of programmed mutagenesis

Programmed mutagenesis is a DNA computation method that implements the sequence-specific rewriting of DNA molecules. Programmed mutagenesis is an *in vitro* mutagenesis technique based on oligonucleotide-directed mutagenesis [4]. Like oligonucleotide-directed mutagenesis, programmed mutagenesis does not mutate existing DNA strands, but instead uses DNA polymerase and DNA ligase to create copies of template molecules with engineered mutations at sequence-specific locations. Every time a programmed mutagenesis reaction is thermal-cycled, a rewriting event occurs. Because the technique relies on sequence-specific rewriting, multiple rules can be present in a given reaction at the same time, with only certain rules being active in a given rewriting cycle. Furthermore, the system's ability to accommodate inactive rules allows it to proceed without human intervention between cycles. We have previously demonstrated the experimental practicality of the key primitive operations required for implementation of programmed mutagenesis systems [28].

There are two main classes of possible designs for the DNA-based string-rewrite systems:

- Decoupled systems, where the sequences of the initial and final strings in each rewrite rule have no similarity or dependence relation between them. Insdel systems discussed in Chapter 1 are an example of a decoupled system.

- Coupled systems, where the sequence of the final string of each successfully executed rewrite rule is strictly dependent on the sequence of the initial string being rewritten. Programmed Mutagenesis is an example of a coupled system.

Coupled and decoupled systems have different sources of sequence specificity. Sources of specificity for both systems include the thermodynamics of DNA hybridization, and secondary structure considerations. In the coupled systems, additional specificity comes from the relation between the sequences of strings being rewritten, and, equally important, from the relation between the sequences of the strings which should not interact. In particular, in programmed mutagenesis systems rewrite rule specificity is in part determined by the number and geometry of mismatches. The geometry of mismatches strongly influences the specificity of the rewrite rules. Other factors involved include temperature of the reaction, concentrations of templates and primers, and enzyme specificity. All these parameters contribute to the dimensions of transition space around a given sequence.

Certain DNA computing systems are known to be restricted in the types of computations they can perform. For example, the generate-and-search methods discussed in Chapter 1 are useful for certain types of combinatorial problems, but these DNA computing methods can not be used to implement general computation [71].

We address the open question of the computational power of a programmed mutagenesis system in Chapter 3 by constructing an encoding of the smallest known Universal Turing machine, Minsky's 7-state 4-symbol Universal Turing machine [44] in the programmed mutagenesis model. A Turing machine is an abstract model of a programmed computer. A Universal Turing Machine is a machine that is capable of simulating any Turing machine, given the description of that machine and the input to the machine [61]. A Turing machine is a string-rewrite system because operation of the machine can be described by a set of quintuplets of the form (*old state, sym-*

bol scanned, new state, symbol written, direction of motion), i.e. as quintuplets in which the third, fourth, and fifth symbols are determined by the first and second. We provide a constructive reduction of a particular Universal Turing machine to programmed mutagenesis, and show how to encode the tape of a Turing machine into a DNA molecule.

2.2 Unary counter

An example programmed mutagenesis system that implements a unary counter is shown in Figure 2-1. The template (I) contains an encoding of a series of symbols XZZZZZ embedded in a noncoding region. The machine is called a unary counter because we can think of the counter as being incremented every time the system is thermo-cycled. We say that the number of symbols other than Z (i.e. X and Y) in the coding region minus one is the current count in the counter. Thus, template I carries the count of zero, since it contains one symbol other than Z. Every mutagenic cycle rewrites another Z into either X or Y, incrementing the counter by one.

The entire region is cloned into a plasmid using Eco RI and Hind III restriction enzymes. The outer primer MLP (machine left primer) is part of the noncoding region and the outer primer MRP (machine right primer) is a part of the plasmid sequence. Each symbol used in the system (X, Y, and Z) is encoded by a 12-nucleotide long sequence of DNA. The actual encodings used for these symbols are shown in Figure 2-2. The bases at which encodings are mismatched are indicated. X and Y both differ from Z by two mismatches and from each other by 4 mismatches. The system was designed such that any oligonucleotide binding with two or fewer mismatches would be able to bind, extend, and be ligated to, and any oligonucleotide binding with more than two mismatches would not be able to interact with the template. Mismatch locations were designed to minimize the opportunity for inappropriate binding and to negate the ability of an oligonucleotide bound with four mismatches to be ligated to. This was possible because of the requirements on the alignment of the oligonucleotides introduced by enzymes used in this experimental system.

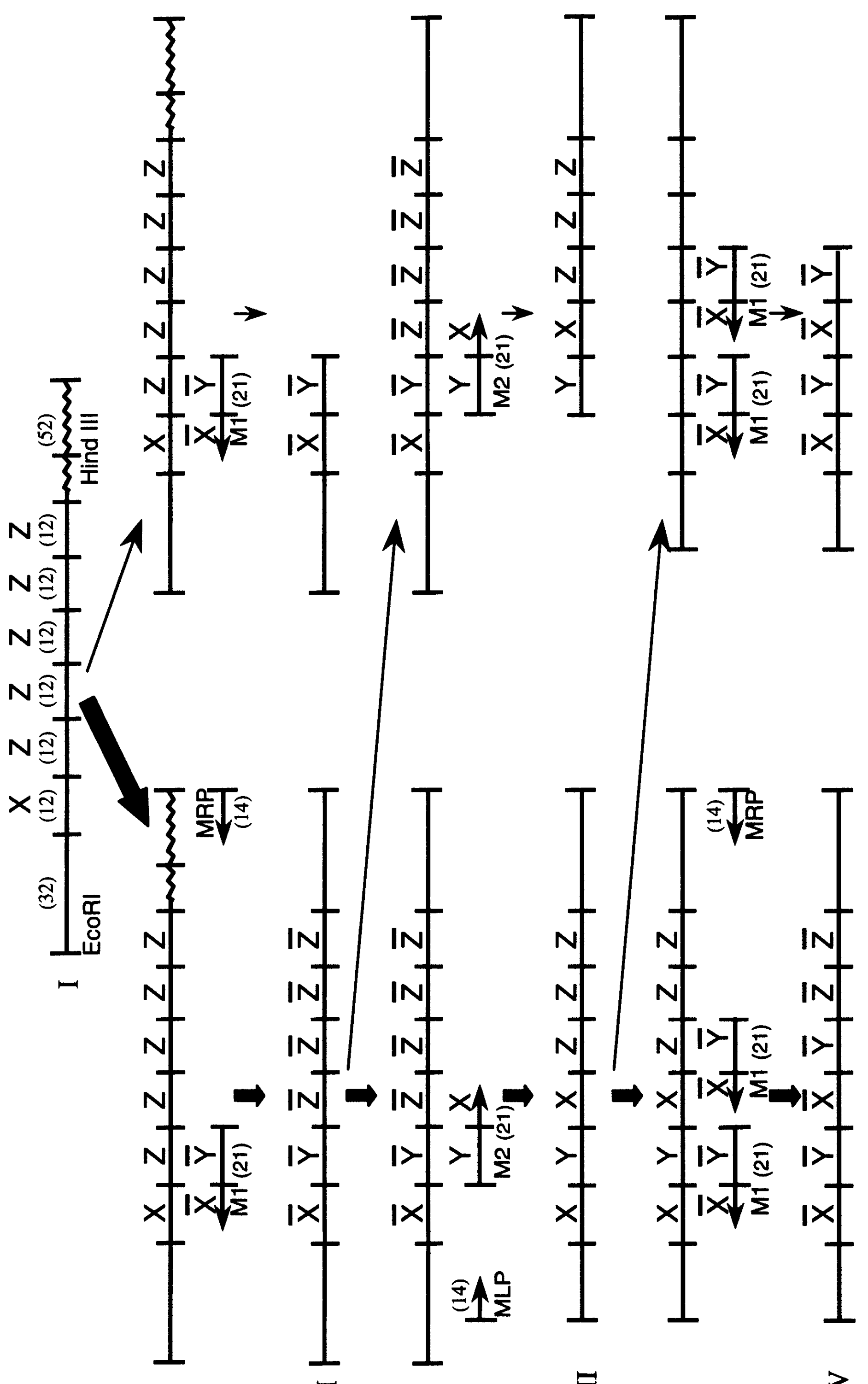


Figure 2-1: Schematic representation of the unary counter. M1 and M2 are mutagenic rule oligonucleotides; MRP and MLP are perfectly matched outside oligonucleotides. Note that a rule incorporated in the previous cycle becomes part of the template for the following cycle. Bold arrows denote the transitions which carry the computation forward. The right side of the figure shows the events which lead to creation of the characteristic bands for each cycle. These characteristic bands are the results of failed ligation events, so named because they represent the result of a failed ligation of the successful extension of the perfectly matched outside primer to the successful extension of the mutagenic rule oligonucleotide.

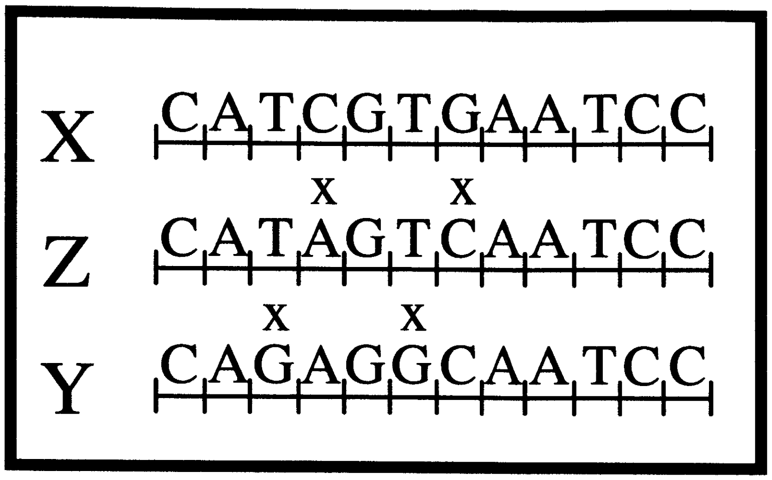


Figure 2-2: Encodings of the symbols used in implementing the unary counter machine in Figure 2-1. All sequences are given in the 5′-to-3′ orientation, and mismatch locations are indicated.

Each oligonucleotide able to anneal in the system is expected to be extended by the polymerase to the end of the available template or until a product of another extension is encountered. When one strand is extended to encounter another oligonucleotide positioned on the template, a ligation event is expected to occur. Ligation is not 100% efficient, and results of failed ligations are expected and are termed characteristic bands of a particular cycle. Such characteristic bands, as well as full-length products are illustrated in Figure 2-1.

Mutagenic oligonucleotide M1 participates in creating a first cycle product (II) that contains a different sequence than the first cycle template (I). This change permits mutagenic oligonucleotide M2 to bind to product II in cycle two, producing another new product III that incorporates M2. Product III contains a sequence that permits oligonucleotide M1 to bind in a yet another location in the third cycle yielding product IV.

Thus a sequence of related novel products (II $\rightarrow$ III $\rightarrow$ IV) is created in a specific order. Outer primers and ligation are used to create full-length products, and all of the enzymes used in the system are thermostable which allows the system to be thermal cycled for progress. The practical feasibility of the underlying specific annealing, polymerization, and ligation operations are examined in the context of a multiple-cycle experiment in Chapter 4.

The challenge in creating an encoding for any programmed mutagenesis system lies in the need to find a set of DNA sequences that has the right mismatch matrix which satisfies the constraints imposed by the desired transition space configuration. It is not a priori obvious that sufficiently complex relationships can be designed. Moreover, to encode target machines, it is often advantageous to expand the encoding to generate a larger mismatch matrix, but one whose requirements can be satisfied by real DNA sequences. In Chapter 3 we discuss in detail how to create a mismatch matrix with requirements that can be satisfied by a set of real DNA sequences.

2.3 A formal model of programmed mutagenesis

As described above, programmed mutagenesis relies on a transition space constraint based on mismatches in rewrite rules to sequence the steps of a program. The number of mismatches is not the only factor determining whether a given rule may bind to and rewrite a given sequence of DNA, but it is the most influential. The transition space constraint is reinforced by three biochemical factors. These three factors are the destabilizing effect of mismatches on duplex stability, polymerase effects, and ligase effects. Under the conditions in our experimental system, for example, polymerase and ligase can not function if mismatches are too close to their action sites. Other factors influencing oligonucleotide's ability to bind a given sequence of DNA, extend a given sequence, or serve as a suitable template for a ligation reaction include mismatch geometry, type of mismatch, enzymes and buffers used, and other biochemical parameters. At the present time it is impractical to try to model in a formal computational model all the parameters influencing the efficiency of binding, extension, and ligation of DNA rewrite rules, in part because insufficient information exists to construct a reliable model. More importantly, mathematical insight would be difficult in an overly detailed model. Therefore, our formal model below uses the number of mismatches to compute an approximation of the transition space of a sequence. We consider the number of mismatches to be the sole determining factor for the ability of a primer (rewrite rule) to bind a given DNA sequence, extend, and to be ligated into a longer strand. In creating an encoding we also take into account requirements of the polymerase and ligase enzymes as to the location of mismatches with respect to the active sites of these enzymes. Thus, we implicitly account for the transition space requirements generated by the use of these particular enzymes. We discuss the biochemical parameters contributing to the dimensions of the transition space in Chapter 4.

An important observation about programmed mutagenesis is that the rules become part of the template for the next cycle. Thus, the rules do not take the familiar form of antecedent $\rightarrow$ consequent, but rather the rules in solution are consequents, searching for their antecedents. In fact, this property is inherent in the underlying biological technique of oligonucleotide-directed (or site-directed) mutagenesis.

The critical role of the transition space constraint in a programmed mutagenesis system now becomes apparent. The consequent of a rule has to be within the transition space of the antecedent of that rule. Equally importantly, the consequent should not be within the transition space of any sequence on the template that is not an antecedent in a rule for which it is the consequent. It is, therefore, intuitive that in modeling a Turing machine we need to explicitly place state onto the tape (and into the rules). Under such an encoding the transition space constraint is reinforced by both the state and symbol encodings. The decision of exactly how to place the state on the tape is an essential part of this proof and is explained in detail in Chapter 3.

We will formally model programmed mutagenesis as a DNA-based string rewrite system in which a single strand of DNA is rewritten in a sequence-specific manner with the use of a set of DNA rewrite rules to produce a single strand of DNA as output. Our formal model of programmed mutagenesis describes rewrite rules that are either 41 or 28 bases long. Also part of the model are the following four assumptions:

1. For a rule of length 41, the allowed mismatch distance is 4 mismatches.

2. For a rule of length 28, the allowed mismatch distance is 3 mismatches.

3. Any DNA sequence for which the distance in mismatches to its target on the template is above the specified number of mismatches will not bind and extend.

4. If only one rule can bind to a spot, that rule executes. If more than one rule can bind to a spot, equal percentages of each rule execute (thus creating parallel branches of computation). Thus, a copy operation occurs only if there are no rules that can affect a rewrite.

It is possible to formulate alternative formal models of programmed mutagenesis that use rules of different lengths and different distance constraints. We have experimentally found above length and distance constraints to be reasonable.

Chapter 3

Programmed mutagenesis systems are universal

We begin this chapter with a formal definition of a Turing machine and a description of the particular universal Turing machine we model. We discuss challenges in encoding a Turing machine in DNA in Section 3.2, give a proof outline in Section 3.3, give the distance metric of the encoding in Section 3.4, demonstrate our encoding scheme in Section 3.5, explain the rewrite rules themselves in Section 3.6, discuss correctness of the encoding in Section 3.7, and, finally, discuss how the methodology used to create this particular encoding can be used to create encodings for other special purpose and universal Turing machines in Section 3.8.

3.1 Turing machines and an example of a universal Turing machine

There are several equivalent definitions of string rewrite systems [44]. Perhaps the most well-known string rewrite systems are Turing machines. A Turing machine is a general model of computation first introduced by Alan Turing in 1936 [61]. The Turing machine model uses an infinite tape as its unlimited memory and finite number of states and symbols to encode and compute a problem. A Turing machine has a head which can move both left and right on the tape, reading and rewriting only the symbol it is currently pointing to. The behavior of a Turing machine is governed by its transition function, which, given the current state and symbol being read, determines the new state the machine will enter, the new symbol to be written on the tape, and the direction of motion of the tape head (left or right). The formal definition of a Turing machine follows.

Definition 1

A Turing machine is the 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{halt})$, where Q, Σ and Γ are all finite sets and

1. Q is a set of states
2. Σ is the input alphabet
3. Γ is the tape alphabet, which includes the blank symbol
4. δ: $(Q \text{ x } \Gamma) \rightarrow (Q \text{ x } \Gamma \text{ x } \{L, R\})$ is the transition function
5. q_0 is a start state
6. q_{halt} is a halt state

A Turing machine begins computation with its head on the leftmost cell of the input string, and proceeds by following the transition function. If it ever enters q_{halt}, it halts. If the head ever reaches the last tape cell on either side, and the transition function indicates moving off the tape, machine appends a single tape cell with a blank symbol on it.

An instantaneous description of a Turing machine is its configuration– the current state of the finite control, the current tape contents, and the current head location. Figure 3-1 shows how the instantaneous description of a Turing machine progresses during a single transition computational step. Corresponding DNA representations are depicted in the bottom portion of Figure 3-1, and are discussed below.

A *universal* Turing machine is a single Turing machine U, with the property that for each Turing machine T which computes a Turing-computable function f, there is a string of symbols d_T such that if the output of T on input x is *f(x)*, then the output of U on input x d_T is also *f(x)* [44].

The smallest known universal Turing machine was described by Marvin Minsky [44], and it has 4 symbols and 7 states. In this machine, Σ and Γ are identical, with 0 playing the role of the blank symbol as well.

Definition 2

Minsky's Universal Turing Machine (MUTM) is the 6-tuple $(Q, \Sigma, \Sigma, \delta, q_1, q_3)$, where

1. Q=$\{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
2. Σ=$\{y, 0, 1, A\}$, with 0 also acting as a blank symbol.
3. δ is given by the transition table in Table 3.1(where the state remains the same unless specified by a number after the /):

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
y	0L	0L/1	yL	yL	yR	yR	0R
0	0L	yR	HALT	yR/5	yL/3	AL/3	yR/6
1	1L/2	AR	AL	1L/7	AR	AR	1R
A	1L	yR/6	1L/4	1L	1R	1R	0R/2

Table 3.1: Transition table of the MUTM. Note that HALT state is achieved only if while the machine is in state q_3 it encounters a 0.

An interesting feature of the MUTM is that the halt state is only achieved when a particular symbol– 0 is encountered while the machine is in state q_3. Otherwise, q_3 behaves like any other machine state.

It is important to realize that this particular machine experiences exponential slowdown in speed with respect to the Turing machine it simulates because MUTM encodes the input to the machine being simulated in unary. Thus, MUTM is not a practical machine in a sense that we are not likely to want to implement this particular machine in a laboratory. However, it is a useful theoretical tool. Because of the limited number of components (states and symbols) in MUTM it is easier to conceptualize and encode than a larger machine. In Section 3.8 we will discuss how to extend the ideas developed for the encoding of this particular machine to a broad class of other universal and special purpose machines.

3.2 Modeling a Turing machine with programmed mutagenesis

There is a number of challenges in modeling a Turing machine within the programmed mutagenesis framework. First, as noted in Chapter 2, the rewrite rules, or more precisely, the consequents of the rewrite rules executed during a particular cycle become part of the template for the next cycle, the property which necessitates explicitly placing the state onto the template and into the rules.

Another important observation about the consequents of the Turing machine rules is that there are no "read" and "write" parts to them. This means that all we can say about the encodings of the rules is that the consequent of a rule has to be within the transition space (radius d) of the antecedent of that rule, and no other sequence. In other words, we can not define which part of a rule passes the information and which receives it. In contrast, in DNA it is a lot more intuitive to think of a rewrite rule as containing the read and write parts.

For example, in the unary counter (Figure 2-1), each rule clearly has a read and write parts. More precisely, each rule has a read only part and a read/write part. In fact, the meaning of the rules used in the unary counter can be summarized as follows:

- M1: **if** the sequence X is present on the template (read part), followed by the sequence Z (read/write part),
  then rewrite the sequence Z into the sequence Y.

- M2: **if** the sequence Y is present on the template (read part), followed by the sequence Z (read/write part),
  then rewrite the sequence Z into the sequence X.

Turing machine rules do not have the convenience of separate read and read/write parts. Therefore, we begin to see that it may be necessary to introduce intermediate symbols to allow transitions to occur in a way more natural for DNA rewrite rules. Thus, we design several DNA rules to execute one MUTM rule. We describe how this is achieved in Section 3.3.

Turing machine rules are also a many-to-one function, which means that several antecedents have the same consequent. This property means that said consequent has to be no more than d away from all its antecedents, and no other sequence. The problem arises because sometimes a symbol is rewritten to itself, and sometimes to another symbol. Sometimes both of these cases are antecedents to the same consequent. In the case where two antecedents lead to the same consequent, and one of these rules involves a symbol that is rewritten to itself, while the other involves a different symbol being rewritten to a symbol other than itself, the consequent may be closer to one of its antecedents than to another. Such an event would create problems because if the symbol on the template is perfectly matched to a symbol in the rewrite rule, the other part of the rule may bind to something that is farther away than permitted, but, because of the mismatches "saved" on the symbol, be able to interact with this wrong site. Thus, it is desirable to have one-to-one rules in our DNA encoding. In order to achieve this property we design our encoding in such a way that we have four DNA replication cycles for every MUTM computational step. Only the first three biological cycles, however, include rewrite events, while the fourth simply copies the template strand using the outside primer. Furthermore, the rules act on alternative strands of DNA ($3'$-to-$5'$ for cycles 1 and 3 and $5'$-to-$3'$ for cycle 2), and we use this property to drive the computation forward.

3.3 Proof outline

We have designed an encoding in our model of programmed mutagenesis to directly simulate the MUTM. There are three key ideas in this construction.

1. In order to explicitly represent the location of the head and the current state on

the tape, we extend the tape alphabet (which, by Definition 2, is $\Sigma=\{y, 0, 1, A\}$) to include q_i/s for every $q_i \epsilon Q$ and every $s \epsilon \Sigma$. We will further expand the alphabet, but this is the essential decision.

We decided to extend the alphabet rather than insert a separate tape cell for the state symbol because the latter format would result in having rules of different lengths for the MUTM rules moving the head to the left and to the right. This is because the right moving rule would only need to take into account the tape symbol immediately to its right, while the left moving symbol must also accommodate the symbol to the left. We have determined that in such an encoding scheme some of the longer rules would inadvertently bind when a shorter rule is intended to bind. This would happen if both of the tape symbols matched those surrounding the current state symbol on the tape. We believe that this may be an insurmountable obstacle to an encoding which seeks not to extend the tape alphabet.

2. We introduce scratch space (#) between each two tape symbols. This allows us to have read and read/write parts of each rule. We use the scratch space to transmit information by first writing to it the information on the new state and the direction of movement (using the q_i/s cell as the read part of the rule). We then use the freshly written information as the anchor for the next step of the rule.

3. We execute each rule of the MUTM by four rewrite rules.

    - read the current state and symbol and save the information about the new state and the direction of the rule in the scratch space; begin the transition to the new tape symbol;
    - finish the transition to the new tape symbol;
    - read the new state info (in the scratch space), begin transition to the the new state/symbol pair;

- finish transition to the the new state/symbol pair, and return the scratch space to its original state (#).

The rules are executed over four biological cycles. Only the first three biological cycles, however, include rewrite events, while the fourth simply copies the template strand using the outside primer. Furthermore, the rules act on alternative strands of DNA ($3'$-to-$5'$ for cycles 1 and 3 and $5'$-to-$3'$ for cycle 2), and we use this property to drive the computation forward.

Spreading one computational step over four DNA rewrite cycles allows us to have one-to-one rules, because each antecedent takes a different path to the consequent. This is achieved by extending the alphabet by a different intermediate symbol for each starting state/symbol pair and the symbol the pair is to be rewritten into as described below. To be precise, we have one-to-one rules for three out of four steps. Step 2 can be and is executed as many-to-one.

The instantaneous description of the DNA encoding of the MUTM at the beginning of each computational step is the single instance of a state/symbol character on the tape, indicating the precise head position and the state of the finite control, and all the other symbols on the tape. We represent states and symbols by nucleotide sequences and enact state transitions by primer extension reactions. The bottom portion of Figure 3-1 depicts the instantaneous descriptions of the DNA representation of a Turing machine before and after a single transition depicted in the top portion of the figure. The four rewrite rules which execute the computational step are indicated on the transition arrow. All possible transitions within the computational step are discussed in Section 3.7 and are illustrated in Figure 3-4.

3.4 Distance metric and alphabet extensions

As discussed above, in addition to the 4 tape symbols $(y, 0, A, 1)$ we introduce symbols q_i/s that are 6 mismatches away from s for all q_i. Also, if q_i/s is to be rewritten according to the rule $(q_i, s) \rightarrow (q_j, s', L/R)$, i.e. if the rule rewrites s to s', then q_i/s

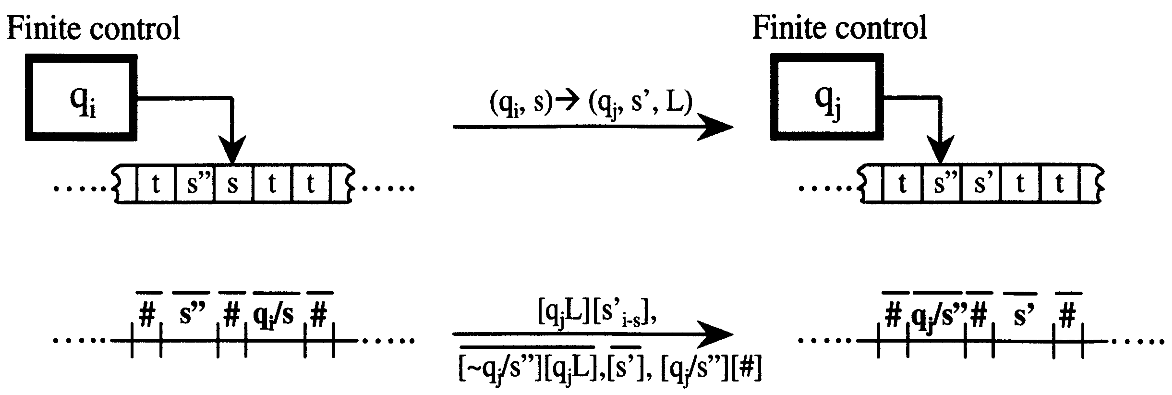


Figure 3-1: Schematic representation of a state transition of the MUTM and the corresponding DNA representations. Indicated on the arrows are the transition rule of the Turing machine and the corresponding DNA rewrite rules necessary to execute the computation step.

is also 6 mismatches away from s'.

We now introduce additional symbols needed to complete the proof. We summarize discussion below by presenting the key distances between tape symbols in Figure 3.2.

- For every transition rule (q_i, s)$\rightarrow$($q_j, s', L/R$), where s may or may not be the same as s', we introduce a new symbol s'_{i-s}, 28 bases long, which is 3 mismatches away from q_i/s and 3 mismatches away from s'. Because of the particular method of encoding used here, s'_{i-s} is guaranteed to be at least 5 mismatches away from any other q_k/s''.

- For every rewrite rule $(q_i, s) \rightarrow (q_j, s', D)$, where $D \epsilon \{L, R\}$, we introduce a new symbol: q_jD, 13 bases long, which is 1 mismatch away from $\#$. Because of the transition table of this particular machine, it turns out there are only 9 symbols in this class, corresponding to the following (new state, direction) pairs: (q_1, L), (q_2, L), (q_2, R), (q_3, L), (q_4, L), (q_5, R), (q_6, R), (q_7, L), (q_7, R). Under the particular encoding used, any q_iD_1 is at least 2 mismatches away from any q_jD_2 if i $\neq$ j or $D_1 \neq D_2$.

- For every rewrite rule of the MUTM $(q_i, s) \rightarrow (q_j, s', D)$, where the next tape symbol in the direction of D is s'', we introduce a new symbol $\sim q_j/s''$, 28 bases long, which is 4 mismatches away from s'' and 3 mismatches away from q_j/s''.

(a)

	s	s'	s''	q_i/s	q_k/t	s'_{i-s}	s'_{i-t}	s'_{k-s}	t_{k-v}	$\sim q_j/s''$	q_j/s''	$\sim q_k/s''$
s	0											
s'	≥ 12	0										
s''	≥ 12	≥ 12	0									
q_i/s	6	6	≥ 6	0								
q_k/t	≥ 6	≥ 6	≥ 6	≥ 6	0							
s'_{i-s}	9	3	≥ 12	3	≥ 5	0						
s'_{i-t}	≥ 12	3	≥ 12	≥ 7	≥ 5	≥ 5	0					
s'_{k-s}	9	3	≥ 12	≥ 5	≥ 7	≥ 4	≥ 5	0				
t_{k-v}	≥ 12	≥ 12	≥ 12	≥ 5	≥ 9	≥ 5	≥ 5	≥ 5	0			
$\sim q_j/s''$	≥ 9	≥ 9	4	≥ 5	≥ 5	≥ 5	≥ 5	≥ 5	≥ 5	0		
q_j/s''	≥ 6	≥ 6	6	≥ 6	≥ 6	≥ 5	≥ 5	≥ 5	≥ 5	3	0	
$\sim q_k/s''$	≥ 9	≥ 9	4	≥ 5	≥ 5	≥ 5	≥ 5	≥ 5	≥ 5	≥ 5	≥ 5	0

(b)

	#	q_iD_1	q_jD_2 for i $\neq$ j or $D_1 \neq D_2$
#	0	1	1
q_iD_1	1	0	2
q_jD_2	1	2	0

Table 3.2: Key distances between tape symbols. Table (a) shows distances for elements of size 28. The distances given are for a rule $(q_i, s) \rightarrow (q_j, s', D)$, where the next symbol in the direction of D is s''. s, s', s'' may or may not be the same. Table (b) summarizes the distances for elements of size 13.

We also guarantee that it is at least 5 mismatches away from any s'''_{k-t}.

3.5 Encoding scheme

We now demonstrate the encoding scheme that allows us to hold to the distance metric above.

We represent scratch space symbols with 9 coding bases surrounded on either side by 2 spacer bases (CC and GC). We employ a spacer so that no mismatch in the coding positions is adjacent to the site of action of either polymerase or ligase. We have found empirically that such a mismatch would prevent the enzymes from working. A # symbol is a sequence of 9 Gs surrounded by the buffer bases. As illustrated in Figure 3.2(b), each of the q_iD's is different from # in one of these 9 positions, and, therefore, different from each other in 2 positions.

The 28-base long symbols are represented by 24 coding bases surrounded, once again, by 2 spacer bases on each side. The relationships between the 28-base long symbols are represented in Table 3.2(a) and Figure 3-2. Directions of the arrows in Figure 3-2 represent the direction of rewrite, and all the arrows represent distance of 6 mismatches. As can be seen in Figure 3-2, encoding of symbol 1 is only related to encoding of A, and has no connection with encodings for either y or 0. We capitalize on this observation in creating the following encoding scheme.

We designate the odd positions of the coding section as characteristic bases of y, 0, and A. We designate the even positions as characteristic of A and 1. As can be seen in Figure 3-3, all odd coding bases for y are As, for 0– Cs, and for A– Ts. Even bases are random, and are the same for y, 0, and A, and different for 1. Thus, y, 0, and A are 12 mismatches away from each other, as are A and 1, but y and 0 are 24 mismatches away from 1.

If a q_i/s is rewritten into an s', we encode it by changing 6 of the characteristic bases of s into those of s'. For q_j/s's that are rewritten to s, we change 6 of the bases that are not characteristic bases of s. Under our encoding, all of these symbols are at least 6 mismatches apart.

If q_i/s is rewritten into an s', we encode s'_{i-s} by changing 3 of the 6 characteristic bases of s within q_i/s into those of s'. For q_j/s which is rewritten into an s, we change 3 of the 6 bases that are not characteristic bases of s within q_i/s back into the characteristic bases of s. Under our encoding there are only 5 pairs of symbols of this type that are 4 mismatches apart. The rest of the pairs are at least 5 mismatches apart. We show in Section 3.7 why even the presence of symbols that are 4 mismatches apart can not lead the system to execute an illegal rewrite.

To encode $\sim q_i/s$, we start with s and change 3 of the 6 bases that are different between s and q_i/s to those of q_i/s and a fourth to a base that is different from the bases used in both s and q_i/s. Under our encoding, all of these symbols are at least 5 mismatches apart.

We used the above algorithm to generate 85 symbol encodings for the 28-base long symbols, and 10 symbol encodings for 13-base long symbols. Note that we do not need to encode the halt state $q_3/0$ or $\sim q_3/0$ because of our readout mechanism

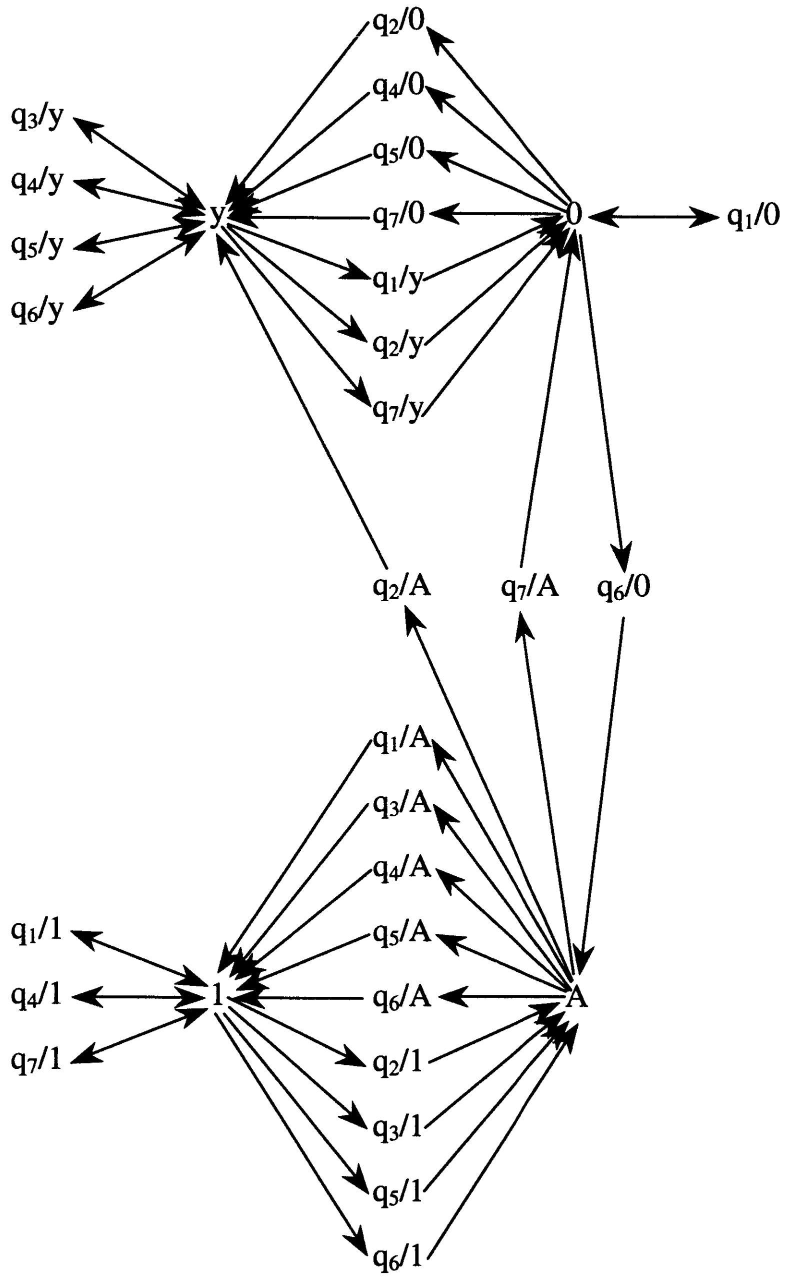


Figure 3-2: Transitions between the symbols used to encode the MUTM. All arrows represent the distance of 6 mismatches and the direction of an arrow represents the direction of the rewrites in the system.

y	C	C	A	T	A	C	A	C	A	G	A	T	A	A	A	G	A	A	A	T	A	G	A	C	A	G	G	C
			X		X		X		X		X		X		X		X		X		X		X		X			
0	C	C	C	T	C	C	C	C	C	G	C	T	C	A	C	G	C	A	C	T	C	G	C	C	C	G	G	C
			X		X		X		X		X		X		X		X		X		X		X		X			
A	C	C	T	T	T	C	T	C	T	G	T	T	T	A	T	G	T	A	T	T	T	G	T	C	T	G	G	C
				X		X		X		X		X		X		X		X		X		X		X		X		
1	C	C	T	A	T	G	T	G	T	C	T	A	T	T	T	C	T	T	T	A	T	C	T	G	T	C	G	C

Figure 3-3: Encodings of and mismatches in the tape symbols of the MUTM.

described in Section 3.6 below. We wrote a simple program to calculate mismatch distances between 85 encodings of length 28. The distance matrix generated by this program, along with the complete list of encodings can be found in Appendix A. We have also verified that any rewrite rule binding off-frame would be more than the allowed number of mismatches away from its target on the template.

3.6 Rewrite rules

We now describe the actual DNA rewrite rules needed to implement the MUTM. First, we have outside primers in the machine that are used to create the full length product by ligating the result of the outside primer extension to the inner mutagenic primer. Second, the rules below are shown in the same orientation regardless of which strand they act on, but we include the $5'$ and $3'$ markers. In total, it takes 101 DNA oligonucleotides to implement the 27 rewrite rules that constitute the transition function of the MUTM.

For every rewrite rule of the TM $(q_i, s) \rightarrow (q_j, s', L)$, where the next tape symbol to the left is s'', we need the following DNA rewrite rules:

1. $5'[q_jL][s'_{i-s}]3'$
2. $3'\overline{[s']}5'$
3. $3'\overline{[\sim q_j/s''][q_jL]}5'$
4. $5'[q_j/s''][\#]3'$

For every rewrite rule of the TM $(q_i, s) \rightarrow (q_j, s', R)$, where the next tape symbol to the right is s'', we need the following DNA rewrite rules:

1. $5'[s'_{i-s}][q_jR]3'$
2. $3'\overline{[s']}5'$
3. $3'\overline{[q_jR][\sim q_j/s'']}5'$

4. $5'[\#][q_j/s'']3'$

Note that rules 2 and 3 execute simultaneously. Also, there are only 4 rules of type 2 (one for each alphabet symbol). And finally, the four rules act over three DNA replication cycles. The fourth biological cycle consists of simply copying the template molecule using an outside primer. This transforms the template into the mode where a rule of type 1 can act on it. The three rewriting cycles plus a copying cycle is the structure which guarantees that the DNA machine can never go back through the computational steps of the MUTM. In Section 3.7 we go through all possible biological transitions within a computational step and demonstrate that while within the cycles that comprise a computational step some random walk through the states of the template is possible, the DNA machine can never go back through computation.

Because we do not know before the computation starts how much tape space it will require, we need rules that will allow us to extend the tape if necessary. We can achieve this with the use of loopout structures. If we say that our template is embedded between unique sequences $[left - wing]$ and $[right - wing]$, then for each $[q_iR]$ we need the rule $3'\overline{[q_iR][0][\#][right - wing]}5'$. For each $[q_jL]$, we need the rule $3'\overline{[left - wing][\#][0][q_jL]}5'$. These rules insert a blank symbol and a scratch space into the template in the direction of motion of the rule. The rules extend the template by one "square", but only when the use of that extra space is required, i.e. when a (right-) left-moving rule is in the process of being executed but encounters the end of the template instead of the blank symbol on the (right) left.

We return now to the peculiarity of the halt state of MUTM. As noted in Section 3.1, MUTM enters the halt state only when it is in state q_3 and is reading the symbol 0. Thus, in our DNA implementation, a simple way to detect when the computation is complete is to sense the appearance of $0q_3L$ on the tape with a molecular beacon. Molecular beacons are hairpin structures with a flourophore and a quencher on the $5'$ and $3'$ ends respectively in which a perfect match for the loop portion of the molecule forces the hairpin to open, thus separating the flourophore from the quencher and creating a fluorescent signal [62]. Notice that we do not need to wait for $q_3/0$ to appear (and in fact we do not even encode this symbol in DNA), because

$0q_3L$ can only lead to the halt state. Thus, our readout mechanism employs molecular beacons to recognize template molecules where the new state is q_3 and the next symbol to be read is 0.

3.7 Correctness

In this section we discuss the transitions in one computation cycle of the MUTM step by step, and consider which rewrite rules can bind at which stages according to our formal model. Figure 3-4 illustrates the discussion below by presenting all possible transitions in a computation cycle. Bold arrows indicate the transitions that carry the computation forward. We consider here a transition that would be effected by a TM rule $(q_i, s) \rightarrow (q_j, s', D)$. Without loss of generality, Figure 3-4 assumes $D = L$.

- At the first rewrite, all possible q_iD's are the same distance away from #, so only the rules which have the component which is 3 mismatches away from q_i/s will bind. But there is exactly one such rule: the one that rewrites q_i/s into s'_{i-s} (and # into q_jD). The other set of rules of the correct sense (5′-to-3′) is the $[q_k/s'''][\#]$ set. But none of these rules can bind now, since the best they can do is to match # and rewrite an s'' into q_k/s''. However, even in this case, the distance is at least 6 mismatches. Of course, the rule that effected the last transition, that is the rule $[q_i/s][\#]$ (or $[\#][q_i/s]$) can bind here. This would lead to no rewrite, and the next transition can only be effected by an outside primer, returning the template to it original state.

- At the second rewrite, the rules that contain the complement of q_jD are the closest to the target sequence on the template, and of those only the one containing the complement of $\sim q_i/s''$ is the allowed distance away. Any rule containing the complement of any q_kD for $j \neq k$ would be at least 6 mismatches away from the target (2 from the scratch space mismatches and 4 from the symbol mismatches). And finally, none of these rules can bind to the s'_{i-s} spot because the distance between any $\sim q_k/s''$ and s'_{i-s} is at least 5 mismatches. At the same

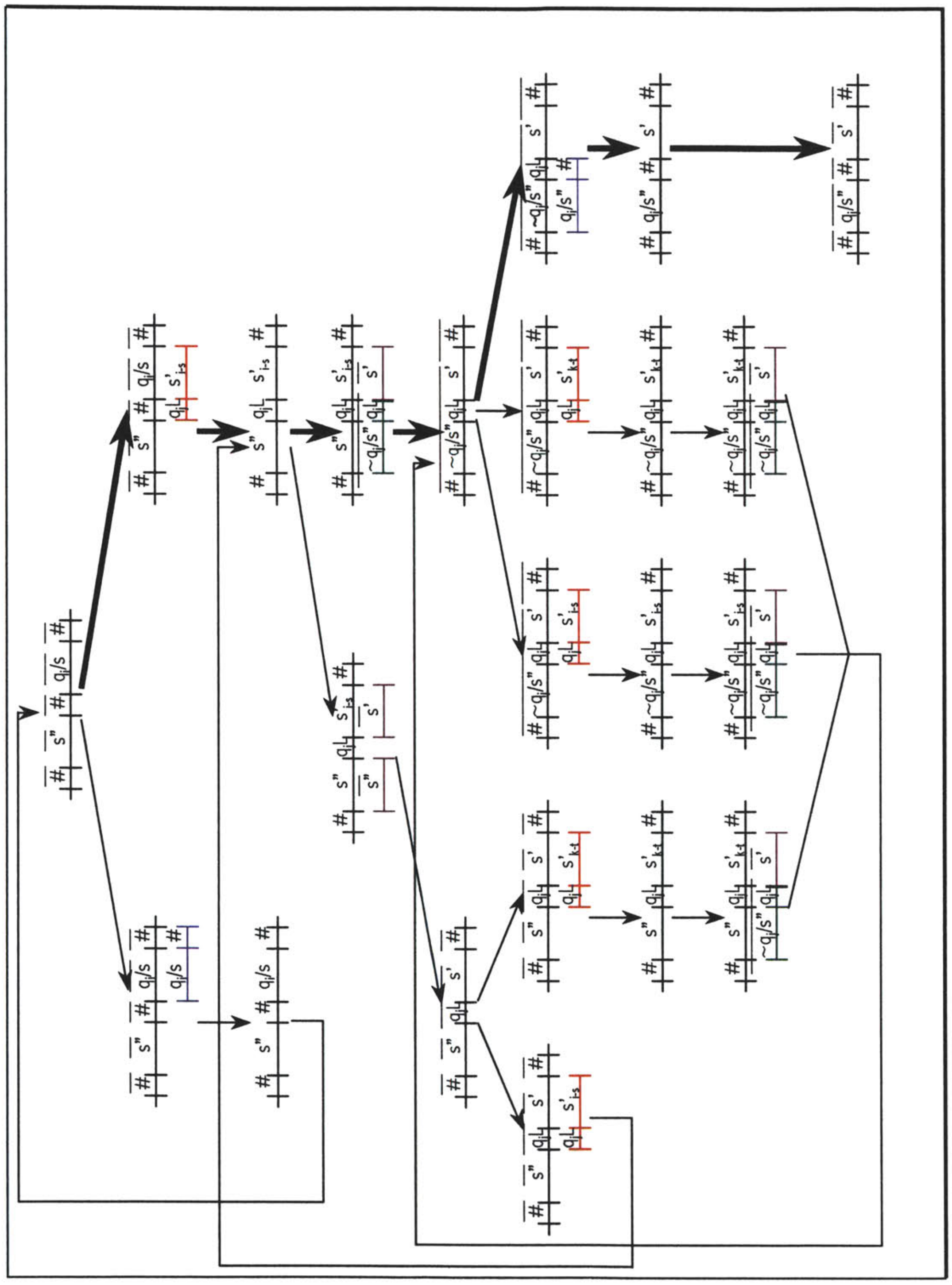


Figure 3-4: Schematic representation of all possible rewrites in a computational cycle. Bold arrows denote the transitions which carry the computation forward.

time, only the complement of s' can bind to s'_{i-s}. The rules containing other s'''_{k-t}'s can not bind because they are of the wrong sense (and thus, it doesn't matter that 5 pairs of these symbols are 4 mismatches apart). Of course, the complement of s'' can bind to s'', thus preventing the rewriting rule from binding. But all that would achieve is enable the first cycle rule to bind again on the 3rd cycle, reproducing the second cycle template again.

- On the third biological cycle, we actually have several rules that can bind. Both the last cycle and the first cycle rules can bind to the same spot– the complement of q_jD. Additionally, other first step rules can bind – those that have the complements of q_jD and of some s'_{k-t} for some k and $t = \{s, s', s'''\}$. Notice that no rule containing q_lD can bind at this stage, since it would be at least 5 mismatches away from the template target sequence (2 in the scratch space and at least 3 in the symbol space). In this situation, if the last cycle rule executes, the process concludes. If, however, a first cycle rule executes, we have on the template (if $D = L$, wlg) sequence $[q_j][\sim q_j/s''][q_jL][s'_{k-t}]$. But this sequence can only be rewritten by the 2nd cycle rules, thus returning the template to the previous state.

As demonstrated above, under our model of programmed mutagenesis and encoding scheme no incorrect rewrite is made, and we ensure forward progress. Thus, we show that above encoding scheme does indeed implement the MUTM, and, consequently, that programmed mutagenesis is a universal model of computation.

3.8 Generalizing the encoding

As we noted above, MUTM is not a practical machine, since it experiences exponential slowdown. We now consider a question of whether the encoding or the encoding methodology used for MUTM has broader applications for encoding other universal or general purpose machines.

The main idea of expanding the alphabet and placing the state symbols onto the tape and into the rules certainly applies to encoding all Turing machines in the programmed mutagenesis framework. We now analyze the other ideas used in this proof to determine how widely applicable they are.

It is important to note that we can increase or decrease the size of DNA rewrite rules for other encodings, but such changes need to be realistic, in that the change in length of the rules should be accompanied by a proportional change in the number of mismatches (our stand-in for the transition space constraint). This proportionality relationship is not linear, since as DNA fragments get longer they are able to tolerate more mismatches with the target sequence and still bind and extend. We do not know exactly what this relationship is, and suggest that an experimental study should be undertaken to map out the relationship.

Using the scratch space symbol (#) between the tape symbols allows for a natural read/write construction of the DNA rewrite rules. Passing information through the scratch space also contributes to our ability to create a separate path from each state/symbol pair to the corresponding symbol into which it is being rewritten. In the MUTM encoding we used scratch space of length 13, which corresponded to the nine new state/direction pairs and four buffer bases. Each state/direction pair differs from the encoding of the # symbol in one of the bases – its characteristic base. Thus, all state/direction pairs differ from each other in two positions – their respective characteristic bases.

This encoding was natural for MUTM, but may not be as easily implementable for other machines. Because the difference in the mismatch distance between an acceptable match and an unacceptable one is only one mismatch, the scratch space encoding should not be too long. Thus, this method of encoding the scratch space symbols would not be well suited for machines with a large number of state/direction pairs. There are, however, heuristics that could be used to keep the length of the scratch space encoding down. One such heuristic takes advantage of the fact that the scratch space symbol is never used by itself as a DNA rewrite rule. It is always used as a part of a rewrite rule which includes a tape symbol.

Some of these tape symbols never appear in rules next to a particular state/direction pair. The distances between the tape symbols themselves vary greatly, as can be seen from the Table 3.2. Some of these distances are far greater than the threshold value for the allowed distance. Thus, two (or, in principle, even three or four) state/direction pairs can share the same characteristic base if it can be assured that no DNA rewrite rule incorporating these state/direction pairs would be able to cause an unintended rewrite to occur.

Software tools can be developed to help in identifying such state/symbol pairs and in creating appropriate encodings. While the encoding scheme would not be as intuitive and transparent as the one used in the MUTM encoding, many larger machines can be accommodated if this heuristic is used. It is worth noting that in machines with larger number of states (and symbols) such non-overlapping state/direction pairs should be easier to identify, even by inspection.

We now consider our encoding method for the tape symbols. Once again, we employ four buffer bases, so the effective encoding space is 24 nucleotides. The size of this fragment can be increased or decreased, but, once again, such changes should vary proportionally with the appropriate changes in the number of allowed mismatches. In the particular encoding of the MUTM, we capitalized on the peculiarity of this particular machine by "splitting" the coding bases into even and odd and designating the odd bases as the characteristic bases of symbols y, 0, and A, and even bases as characteristic of A and 1 (Figure 3-3).

The original template has the starting state/symbol pair on it. From that point on, a state/symbol pair appears on the template because the next symbol in the direction of the rule is rewritten to it. Thus, we can say that each state/symbol pair is created, or comes from, the symbol in that pair. Furthermore, when the next rule is executed, the state/symbol pair is eventually rewritten into a new symbol, which may or may not be the same as the symbol in the state/symbol pair.

Our encoding places the state/symbol pair half way between the old and new symbols, at six mismatches away from each. If the old and new symbols are the same, the state/symbol pair is placed six mismatches away from the symbol by changing the bases which are not characteristic bases of the symbol. Similarly, the intermediate new symbol encodings (s'_{i-s}) are half way between the state/symbol pair being rewritten and the new symbol.

Clearly, the above scheme is tied closely to both the number of allowed mismatches and the particularities of the relationships between the symbols in the MUTM. However, it is worth noting that similar relationships should be easily identifiable in other machines. Furthermore, in machines with a larger number of symbols the number of transitions between any two symbols, in general, should be smaller.

In a less "crowded" field, it may be possible to have smaller distances in mismatches between two symbols while still maintaining the same mismatch distance with the intermediate symbol. For example, capitalizing on the fact that DNA alphabet consists of four bases, it could be possible to have a distance of five mismatches between a state/symbol pair encoding and the encoding of the symbol it is being rewritten to, but still maintain the three mismatch distance from each of these encodings to the intermediate new symbol encoding. Again capitalizing on the fact that the DNA alphabet consists of four bases, it is possible to have up to four different symbols share the same set of characteristic bases.

Because of the variety of properties exploited to create this particular encoding of MUTM, it is not immediately obvious how to describe in mathematical terms the large class of other universal and special purpose machines which can be encoded using this technique, or the variations of this technique described above. It appears likely that machines in which the numbers of states and symbols is on the same order of magnitude would be better susceptible to this method of encoding because of the need to balance the number of mismatches and lengths of each part of a DNA rewrite rule.

As stated above, MUTM experiences exponential slowdown with respect to the machine it is simulating. An open question remains of whether there is a universal Turing machine which does not experience such slowdown, and can still be encoded in DNA within the programmed mutagenesis framework. An ideal machine would have a manageable number of states and symbols, say less than 15 of each. Such a machine would still be small enough that relationships between states and symbols can be easily discovered, understood, and exploited in constructing an encoding.

Chapter 4

Experimental Results and Analysis

We have conducted a series of experiments to establish efficiency of the basic operations of programmed mutagenesis in the context of the unary counter machine depicted in Figure 2-1.

Our preliminary calculations show that programmed mutagenesis systems can operate at speeds of $\sim$4 x 10^{13} polymerization operations per second per unit of polymerase, where an "operation" is polymerization of an entire DNA strand[1]. These systems are also expected to exhibit storage density of $\sim$5 x $10^{-6}\mu^3$/bit, where a bit is a computational symbol, not a base of a DNA duplex; and reduced power requirements of $\sim 10^{16}$ strand rewrite operations/J [13].

In this chapter we discus the two-enzyme system we employ which permits programmed mutagenesis to proceed in a single reaction. We then describe the sources of rewrite rule specificity and discuss how the sequences used in the unary counter machine were chosen. Finally, we present experimental data on the efficiency of the unary counter operation.

[1]Polymerization rate for Vent DNA polymerase (polymerase used in our experimental system) at 45°C (the temperature at which our experimental system was operated) was determined to be $\sim$ 10 nM dNTPs/min at the enzyme concentration of 2.7 U/ml [31]. Taking the strand length to be 1000 bases, we obtain the number above

4.1 Enzyme and Buffer Choice

In its native configuration DNA is a double helix, where the two strands are joined together by the hydrogen bonds between the bases (A, C, G, and T). Bases form stable pairs A-T and G-C. All other base pairings are considered mismatches. Mismatches greatly differ in stability, with G-T being almost as stable as a standard base pair, but all are less stable than the perfect match. DNA strands have polarity. The phosphate-sugar backbone to which the bases are attached is directed from the $5'$ to the $3'$ position of the sugar ring. New bases are added to the $3'$ end of the strand and two strands in the double helix are antiparallel.

The strict A-T, G-C pairings provide the basis for duplicating a DNA strand, since by looking at one strand we can reproduce the sequence of its compliment. DNA replication requires a starting point called a "primer" that is a short DNA molecule which binds to the old strand and creates a stable $3'$ end from which the synthesis of the new strand can proceed.

While living cells employ complex enzymatic machinery to "open" DNA helix in order to begin replication, we use thermal denaturation to achieve the same goal. Unfortunately, most enzymes are thermally deactivated at the temperature required for DNA denaturation. Therefore, in order to avoid having to add enzymes during each step of the computation we employ thermostable enzymes.

We have developed an enzyme system that permits a mutagenic oligonucleotide to be embedded in a newly synthesized DNA strand [28]. As shown in the first cycle of Figure 2-1, in this system a mutagenic oligonucleotide serves as a primer for DNA polymerase on its $3'$ end, and accepts a DNA ligation event on its $5'$ end. In our system these events occur in the same reaction at the same temperature.

The two enzymes that we use in our system are Taq Ligase and Vent Polymerase. Taq Ligase [7] has the virtue of being the only commercially available thermostable ligase. Vent exo^+ Polymerase [7] does not have $5' \rightarrow 3'$ exonuclease activity, and does not unacceptably strand displace at 45^oC. In order to prevent $3' \rightarrow 5'$ proofreading of mutagenic oligonucleotides we manufacture these with sulfur instead of phosphorus linkages on the last four bases. These phosphothioate linkages render the oligonucleotide extendible, but not degradable. Vent *exo*$^-$ Polymerase is similar to Vent, but lacks the $3' \rightarrow 5'$ proofreading function and strand displaces more then Vent *exo*$^+$.

In order for a two-enzyme system to function, both enzymes must function efficiently in a single buffer. Taq DNA Ligase requires NAD as a cofactor. We constructed a custom buffer by adding 10 mM of NAD to 10X Thermopol Vent buffer (hence forth called Vent-NAD buffer). This is the same concentration of NAD as is present in the Taq Ligase buffer. We then tested the efficiency of Vent *exo*$^+$ and *exo*$^-$ DNA polymerases and Taq DNA Ligase in Thermopol buffer, Vent-NAD buffer and Taq Ligase buffer. Taq Ligase buffer and Vent-NAD buffer allow 100% ligation of the control Bst I cut lambda DNA, while Thermopol buffer alone allows only incomplete ligation. However, Taq Ligase buffer does not support efficient polymerization, and thus we chose 10X Vent-NAD buffer for all further experiments.

We hypothesized that reducing the rate of strand extension would help increase the probability of successful ligation events in our system. A molecule of Vent DNA Polymerase extends DNA in increments of 6-7 bases at a time [31]. It then "falls off" the template, and searches for another open $3'$ end to extend. Molecules of Taq DNA Ligase are also constantly searching for a suitable target. We theoretized that once polymerase encounters a properly aligned downstream oligonucleotide, there would be competition between the two enzymes for the ligation site. We thought that it is possible that even though the polymerase cannot extend the $3'$ end any further, it still recognizes the $3'$ end as a possible site of action. Thus, having an unligated $3'$ end next to another strand may enhance the ability of Vent polymerases to displace the other strand.

We experimentally found that excessive amounts of polymerase reduced the amount of ligation product in our system. We tested four different concentrations of the polymerase (1U, 0.25U, 0.125U, and 0.05U per reaction) and concluded that the highest efficiency is achieved with 0.25U of polymerase and 40U of ligase per 10 μl reaction (data not shown).

4.2 Sources of rewrite rule specificity

As discussed above, specificity of the rewrite rules is determined largely by the transition space constraints in the system. In addition to mismatch distance constraints, other sources of specificity are mismatch geometry, processivity of the enzymes, and thermodynamic parameters such as relative amounts and concentrations of template and primers in the reaction, salt concentration, time reaction is allowed to proceed, and reaction temperature. When we began our research, there existed reliable information about the thermodynamics of a perfectly matched DNA duplexes [10], [54], but information about the thermodynamics of mismatched DNA duplexes was less reliable [1], [66]. Through our early experiments we have acquired empirical data and developed intuition regarding mismatched duplexes.

With the appearance of BIND [21], we were able to relate our empirical observations and experiment results to a theoretical prediction. For instance, BIND determined that in our early experiments there was a very narrow T_m difference between oligonucleotides binding in a correct spot and those binding inappropriately. This explained the inappropriate products we were observing in that system.

The DNA sequences for the system we are testing have been selected using the SCAN program [22] to search a large sequence space constrained by chosen mismatch geometry.

SCAN chooses sequences that have optimum annealing properties, lack harmful secondary structure, and do not form primer dimers. We chose fairly strict thermodynamic constraints in order to prevent inappropriate binding of the rule oligonucleotides, as well as any undesirable interaction between primers. Nevertheless, the search space remained too large, and needed to be further constrained. As mentioned above, we chose to constrain that space by defining the geometry of mismatches for the rewrite rules.

The greatest constraint on the oligonucleotide rules search space is placed by the choice of the geometry of mismatches. Mismatches drastically change the thermodynamic characteristics of the oligonucleotides. The characteristics depend on:

1. The particular mismatch used. For example, a C-A mismatch greatly strains the helical structure around itself, while a G-T mismatch has almost no effect on neighboring DNA structure [74].

2. The base pairs surrounding the mismatch [67], [54].

3. The position of the mismatch within the oligonucleotide. Mismatch too close to the $5'$ end could destabilize ligation, while one too close to the $3'$ end may disturb extension. To test our intuition and to collect more empirical data we designed a number of oligonucleotides. These primers were homologous to the elements from our early system and had mismatch structure and several other characteristics of each of the proposed mismatch designs. We used these primers to test the putative extension and ligation efficiencies of the proposed designs. We then used SCAN [22] to search for optimal oligonucleotides with the given mismatch geometry and a common set of thermodynamic constraints. It is interesting to note that the first design we proposed is the one we implemented in the laboratory.

4. The positions of mismatches with respect to each other. It is reasonable to expect that two mismatches right next to each other would have less of an effect on the stability of the duplex then would those same two mismatches located some distance away from each other.

The third point deserves further explanation. Polymerase and ligase enzymes have their own requirements for how stable the site of action needs to be in order for the enzyme to catalyze a reaction. In our particular system, Taq DNA Ligase is extremely discriminatory with respect to the required stability of its site of action. In the experimental conditions under which we operate our system, mismatches within the first four bases of the $5'$ end of the primer being ligated to are not tolerated. Vent DNA Polymerase allows mismatches that are as close to the $3'$ end as three bases. However, these are only tolerated if there is no other mismatch in the immediate vicinity.

Having spent much time in the laboratory working with our early system, we acquired certain intuition regarding mismatch geometry. Based on that, we proposed a design. We also investigated three of its close relatives.

To test our intuition and to collect more empirical data we designed a number of oligonucleotides. These primers were homologous to the elements from our early system and had mismatch structure and several other characteristics of each of the proposed mismatch designs. We used these primers to test the putative extension and ligation efficiencies of the proposed designs. We then used SCAN [22] to search for optimal oligonucleotides with the given mismatch geometry and a common set of thermodynamic constraints.

While the original design of the rewrite rule oligonucleotides proved to work according to specifications, our originally chosen left framing sequence proved to have some homology to the native DNA sequence found on the plasmid into which we cloned the unary counter, as well as to have an ability to form some internal hairpins. These properties necessitated choosing a different left framer sequence and recloning the unary counter machine into a different plasmid.

4.3 Experimental Results

We now turn to the practicality of the primitive operations of a programmed mutagenesis system. We have constructed the unary counter machine shown in Figure 2-1, and have operated it through three cycles to gather efficiency data.

This experiment was the culmination of five years of experimental work. We developed the unary counter machine through four generations, each time improving the design based on the lessons learned in the experiments with previous generations of the machine. We summarize the most important experimental results obtained before moving on to the most recent efficiency results.

As discussed in Chapter 2, failed ligations result in characteristic products. We designed the system such that all these characteristic products have unique and easily distinguishable lengths. We use the appearance of these unique length products to judge whether a cycle of mutagenesis has indeed taken place. Because all these products have unique lengths, and the appearance of characteristic products of cycle n always precedes the appearance of characteristic products of cycle $n + 1$, we did not feel it was necessary to clone and sequence these products to definitively verify their identity. A more complete study would perhaps undertake this cloning effort in addition to the characteristic length verification.

4.3.1 Primitive Operations of Programmed Mutagenesis

In order to demonstrate the viability of programmed mutagenesis as a model of computation, we needed to demonstrate experimentally that it is possible to create full-length product DNA molecules that have embedded rewriting, make later sequence changes to depend on earlier sequence changes, and have multiple oligonucleotides be active in close proximity on a template sequence. These three basic components are present in all programmed mutagenesis systems. All experiments summarized in this section were performed as previously described in [28].

The ability to create molecules with embedded rewrites is at the key feature of programmed mutagenesis. The basic step of any string-rewrite computational system is the creation of a new string based on a template string. In programmed mutagenesis systems this step is accomplished by incorporating a mismatched oligonucleotide into the newly-synthesized DNA strand.

To demonstrate that DNA molecules can be created with internal rewriting events we designed an experiment based on the first cycle reaction of the earlier generation of the unary counter. This system consists of a perfectly matched upstream oligonucleotide MRP and a mismatched downstream oligonucleotide M1. We experimentally showed that primitive operations of programmed mutagenesis, extension and ligation of a mismatched oligonucleotide primer, can occur in a single buffer at a single temperature [28].

The fidelity of a programmed mutagenesis computation depends on the proper incorporation of mutagenic oligonucleotides. Suppose in a given computation the question posed is "Does this computation result in s3?" Now suppose that the set of rewrite rules for this computation includes s2 $\rightarrow$ s3, but not s1 $\rightarrow$ s3, or any other rules which allow s1 to be rewritten. Suppose further that the correct computation terminates with the incorporation of an oligonucleotide rule that rewrites the string into s1. Thus, the answer to the question posed for this computation should be "no." However, if s1 presents a suitable biological template for the execution of s2 $\rightarrow$ s3 rule, this rewrite rule will be incorporated, and it would appear that the result of the computation is "yes." In other words, if the rules are not incorporated strictly sequentially, no claims about correctness of computation can be made.

In case of the unary counter, in order to establish programmed sequential incorporation, we needed to show that the second cycle product appears iff the first cycle has executed successfully. We designed an experiment based on the second cycle reaction of the earlier generation of the unary counter without the left outside primer. We have shown that the second cycle rule oligonucleotide binds in the appropriate location when the first cycle rule oligonucleotide is present in the reaction, and not at all when there is no first cycle rule oligonucleotide in the reaction. Some nonspecific binding was detected at the temperature below the predicted T_m of the second cycle rule oligonucleotide in the correct alignment, but the nonspecific binding was completely eliminated at T_m [28]. We will discuss the influence of concentration of reaction components, as well as the that of the presence of the outside primers in the reaction in subsection 4.3.2.

In order to demonstrate the flexibility of programmed mutagenesis we needed to show that two oligonucleotide rules positioned on the template next to each other can ligate together and extend to the end of the template. This configuration of oligonucleotides is expected to occur when one oligonucleotide binds to the previously rewritten section of the template, while another binds to an adjacent mismatched sequence to be rewritten, as in the third cycle of the unary counter, for example. It is reasonable to suspect that if the mismatched oligonucleotide is downstream of the perfectly matched one, the former may anneal earlier and extend over the binding site of the latter, thus preventing the latter from working.

We experimentally validated that the system consisting of the perfectly matched oligonucleotide immediately upstream of a mismatched oligonucleotide produced full-length products that included both of the oligonucleotides in the reaction. The test showed that the upstream oligonucleotide can ligate to the downstream mismatched oligonucleotide, albeit with a slightly lower efficiency than in the case of two oligonucleotides whose binding sites are separated by some distance [28].

4.3.2 The Role of the Outside Primers in a Programmed Mutagenesis System

In programmed mutagenesis systems outside primers are intended simply as tools to enable production of full-length products on all cycles. This function is important, since often the site of action of the next mutagenic oligonucleotide falls in the area of template upstream of the currently active mutagenic primer. For example, in the unary counter, as can be seen in Figure 2-1, in the absence of MRP, the second cycle mutagenic oligonucleotide M2, would not be able to bind and extend.

Almost from the beginning of our experimental efforts, however, we noticed that sometimes the outside primers also play another role. They seem to prevent a rule oligonucleotide of the same sense (binding in the same orientation) from binding in undesirable locations. In case of the unary counter, we observed that the presence of MLP in the reaction prevented M2 from binding in the positions where M2 was more than 2 mismatches away from the sequence on the template. This effect was observed at a variety of temperatures and concentrations of rule and outside oligonucleotides, in fact persisting over an order of magnitude range of concentrations.

Experiment design was as follows: each reaction contained ~0.2 μM of double stranded template DNA, Taq thermostable ligase (40 U), Vent thermostable polymerase (0.25 U) in 1X Vent-NAD buffer, with the variety of concentrations of inner rule oligonucleotides and outside oligonucleotides. Concentration of the outer oligonucleotides was always ten times that of the inner rule oligonucleotides. These reactions were thermal-cycled for 1 minute at 94°C, and 30 minutes at target temperature for two cycles. We compared reactions containing MLP to those not containing MLP.

We also compared negative control reactions, those lacking M1, with and without MLP. Negative control reactions contain the same components as second cycle reactions, with the exception of M1, first cycle mutagenic primer. In the absence of M1, no product labeled II in Figure 2-1 is supposed to be produced, and no second cycle product is therefore possible. We expect negative control lanes to contain no product. All the negative control reactions discussed in this chapter are set up as described here.

Since we were interested in the effect MLP seemed to have on the binding of M2, M2 was ^{32}P end-labeled. Since failed ligation events produce characteristic length bands, which are the same length bands as those produced in the reactions lacking MLP we could directly observe the effect MLP had on the reactions by comparing bands present in the lanes containing MLP to those in the lanes not containing MLP.

Figure 4-1 shows result of one representative experiment. This particular experiment was performed at 45^oC and contained 0.01 μM of the inner rule oligonucleotides.

As can be seen in Figure 4-1, lanes which do not contain MLP exhibit a number of bands spaced at regular intervals of about 12 base pairs apart. These correspond to M2 binding to all the available positions on the template. The bold band of approximately 112 bps in the cycle 2 lane (cartoon 2 in Figure 4-1) corresponds to M2 binding in its appropriate spot (with two mismatches), while the fainter bands (cartoons 4-7 in Figure 4-1) represent binding to all the locations marked by Z in Figure 2-1 (with four mismatches). Note that all these bands are present even in the negative control lanes.

In contrast, in the lanes containing MLP no inappropriate binding is detectable. Cycle 2 lanes exhibit only the 112 bp band (cartoon 2 in Figure 4-1) and the full-length band (cartoon 1 in Figure 4-1), which is the result of the product of the extension off of MLP ligating to the 112 bp product. Negative control lanes exhibit no bands whatsoever.

Estimated T_m of M2 binding with four mismatches is 20^oC lower than that for binding with two mismatches. Consequently, equilibrium between bound and unboud states for M2 binding with four mismatches is shifted significantly towards the un-

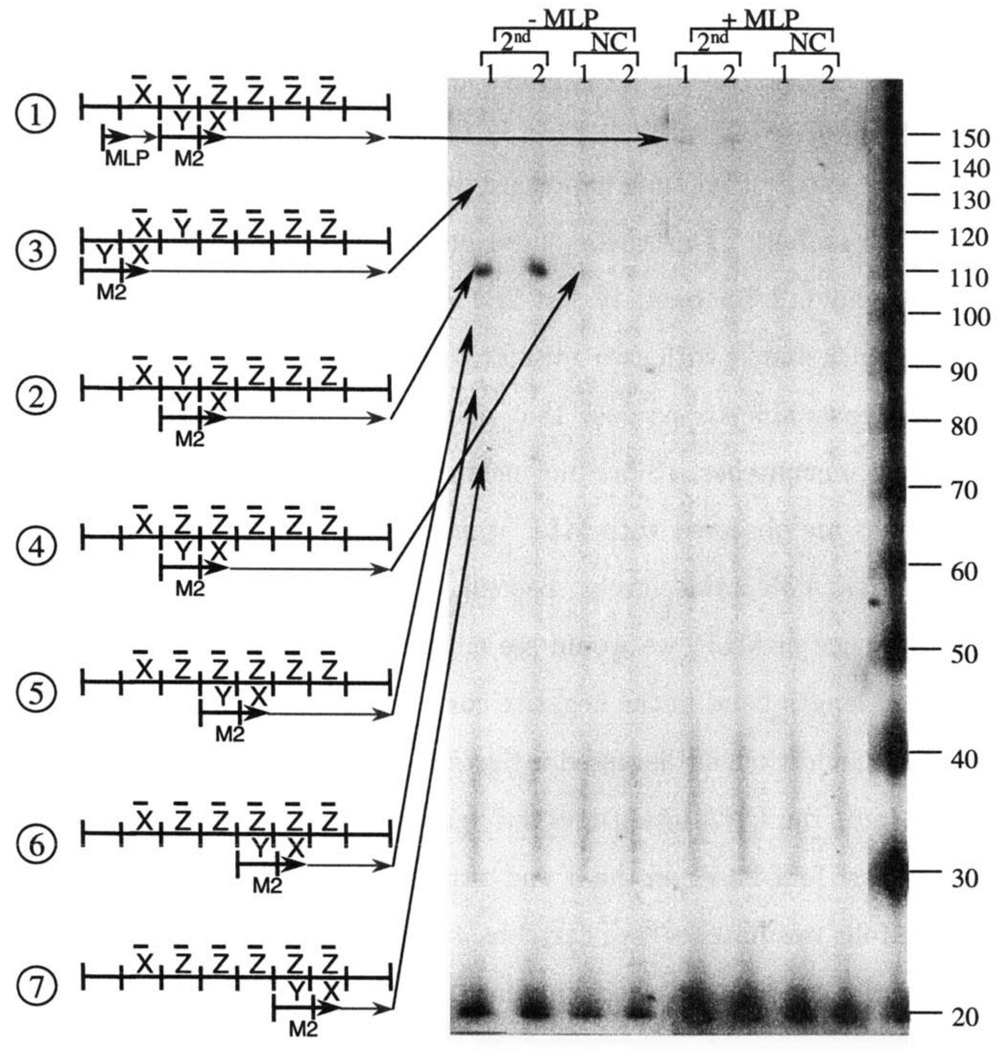


Figure 4-1: The role of MLP in preventing inappropriate binding by M2. Oligonucleotide M2 is end-labeled with ^{32}P. Cartoons represent the products contained in the corresponding bands. Each reaction is repeated twice (labeled 1 and 2). Second cycle (2^{nd}) and negactive control (NC) reactions were run in the absence (-MLP) and presence (+MLP) of MLP. Negative control reactions contain the same components as second cycle reactions, with the exception of M1, first cycle mutagenic primer. In the absence of M1, no product labeled II in Figure 2-1 is supposed to be produced, and no second cycle product is therefore possible. We expect negative control lanes to contain no product.

bound state, as compared to the equilibrium of M2 binding in its appropriate place with two mismatches. Thus, the product of extension off of the outside primer occurs more readily through the possible binding sites, resulting in the observed "cleaning" phenomenon.

Notice that we can not explain the absence of inappropriate bands in the lanes containing MLP by the theory that such bands are simply ligated into the full-length product when MLP is around. This theory does not work for two reasons. First, we can still observe the 112 bp band in the 2nd cycle lanes containing MLP. If the oligonucleotide which binds with two mismatches is not ligated to with 100% efficiency, than it is reasonable to suppose that neither would the oligonucleotides which bind with four mismatches. Since no incomplete ligation products for the four-mismatch bindings are observed with MLP, it is reasonable to assume that such bindings do not occur at detectable levels. Second, if inappropriate bindings were happening in the presence of MLP, we would see either the characteristic bands for such bindings or a full-length band in the negative control lanes with MLP. Neither of these are observable. In addition, as discussed in Section 4.3.1, under the experimental conditions in our system, Taq DNA ligase does not ligate to a mutagenic primer where a mismatch is in the first four bases on the $5'$ end of the primer, as would be the case for M2 binding with four mismatches (see cartoons 4-7 in Figure 4-1). Therefore, we conclude that under the conditions tested, presence of MLP prevents inappropriate binding of M2.

We repeated this experiment at variety of concentrations for 45, 47.5, and 50^oC (data not shown). At 45^oC, the effect persisted across an order of magnitude difference in concentrations of primers. It is important to note that the presence of an outside primer by itself does not guarantee absence of inappropriate binding. For example, we found that at target temperature of 45^oC, with the concentration of inner oligonucleotides of 0.05 μM, inappropriate binding still persisted. However, it was eliminated by raising target temperature of the reaction to 50^oC (data not shown).

(a)

	1st cycle	1st cycle	1st cycle
	full length, %	characteristic fragment, %	ligation efficiency, %
0.01uM rewrite rules	1.06	14.53	6.78
0.02uM rewrite rules	0.53	5.4	8.99
0.03uM rewrite rules	0.46	3.21	11.63

(b)

	2nd cycle	2nd cycle full length,
	full length, %	% of 1st cycle full length
0.01uM rewrite rules	0.07	6.68
0.02uM rewrite rules	0.07	12.67
0.03uM rewrite rules	0.04	8.76

(c)

	3rd cycle	3rd cycle characteristic fragment,
	characteristic fragment, %	% of 2nd cycle full length
0.01uM rewrite rules	0.07	51.85
0.02uM rewrite rules	0.05	34.96
0.03uM rewrite rules	0.05	66.98

Table 4.1: Measured and extrapolated efficiency data. Table (a) shows data for cycle 1, table (b)– for cycle 2, and table (c)– for cycle 3.

4.3.3 Efficiency of the Unary Counter

Figure 4-2 shows all the bands we quantitated to asses the efficiency of the operations of the unary counter. The experiment was performed at three different concentrations of the inside and outside primers. Efficiency data gathered in the experiment is summarized in Table 4.1 and is illustrated in Figure 4-3.

The cycle reactions contained 0.01, 0.02, or 0.03 μM of M1 and M2 oligonucleotides, 0.1, 0.2, or 0.3 μM of outer primers respectively, $\sim$0.2 μM of double stranded template DNA, Taq thermostable ligase (40 U), Vent thermostable polymerase (0.25 U), and 1X Vent-NAD buffer. These reactions were thermal cycled for 1 minute at 94^{o}C, and 30 minutes at 45^{o}C for the indicated number of cycles. Each reaction was repeated twice, to minimize the impact of experimental error on the data.

To estimate the amount of products designated by II, III, and IV in Figure 2-1 we ^{32}P 5$'$ end-labeled M1 (for II and IV) and M2 (for III). Failed ligations yield products of characteristic lengths (as illustrated in Figure 2-1). The results of the reactions were run on polyacrylimide denaturing gels, and bands were quantiated on a phosphorimager.

Figure 4-4 shows the key portion of our experimental data. This data was collected

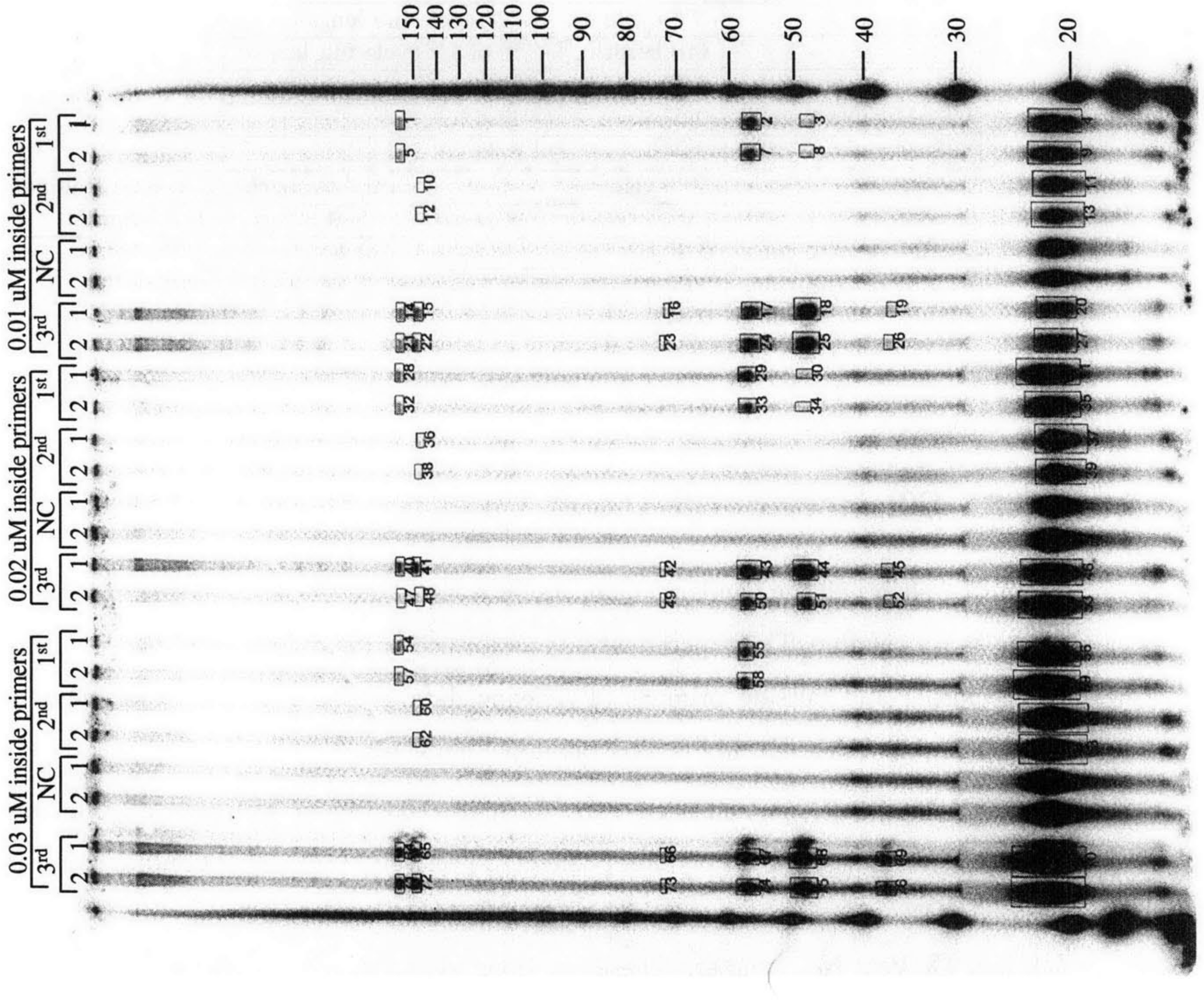
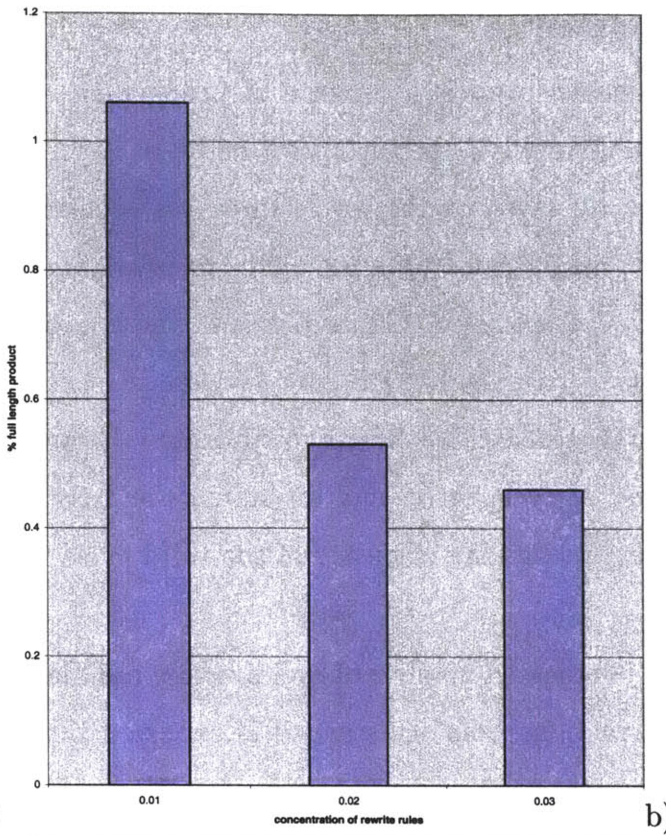
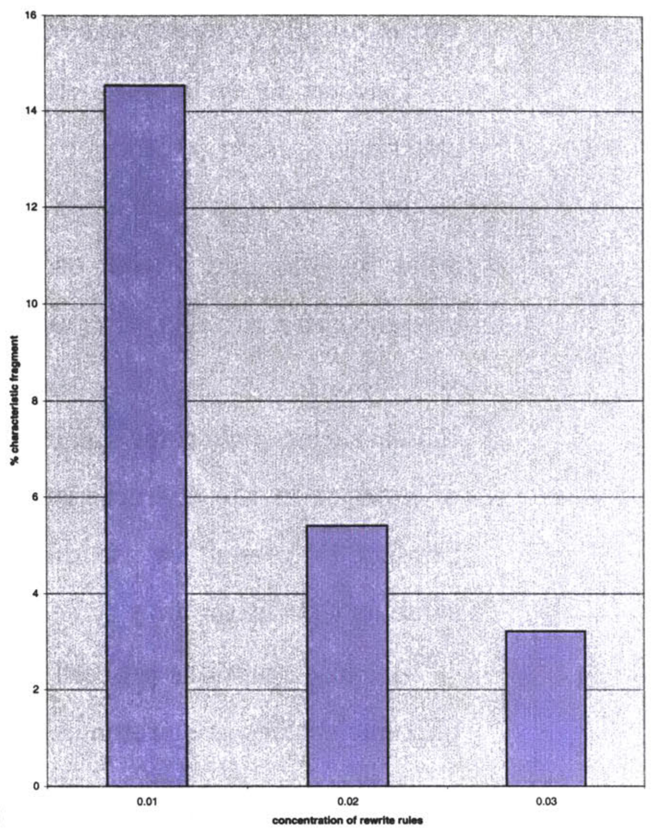


Figure 4-2: Operation of the unary counter machine through cycles 1, 2, and 3, plus negative control (NC). Negative control reactions contain the same components as second cycle reactions, with the exception of M1, first cycle mutagenic primer. In the absence of M1, no product labeled II in Figure 2-1 is supposed to be produced, and no second cycle product is therefore possible. We expect negative control lanes to contain no product. Notice that the gel appears as a mirror image of its normal orientation. Oligonucleotide M1 is end-labeled with ^{32}P in Cycles 1 and 3, while M2 is labeled in Cycle 2 and negative control. All the bands that were quantitated on the phosphoroimager are marked and numbered.

a)


b)


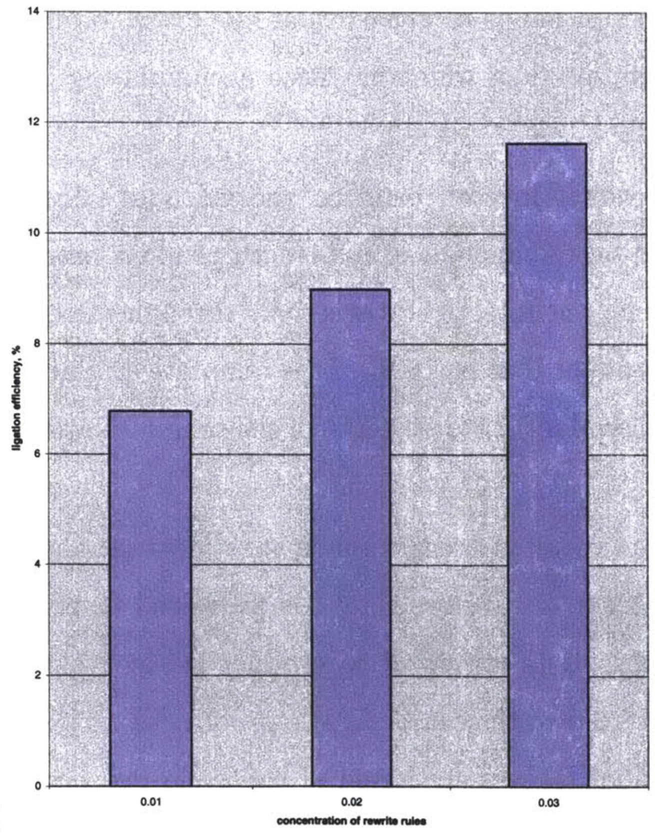
c)


Figure 4-3: Bar graph representation of the efficiency data from cycle 1. Figure (a) shows exponential decline in full length product figure (b)– exponential decline in characteristic length product, and figure (c)– linear increase in ligation efficiency, all with respect to the concentration of rule oligonucleotides in solution.

at the middle concentration of the primers.

Because in cycle three the full-length band is a mixture of product IV and a shortened version of product II (as illustrated by the cartoons 7 and 6 in Figure 4-4, respectively), it is impossible to directly ascertain that cycle three has occurred, or what the efficiency of that cycle is, from the full-length product. However, both the characteristic product of cycle three (cartoon 8 in Figure 4-4) and the coloring of the characteristic product of cycle two (cartoon 10 in Figure 4-4) are present, and indicate that cycle three has indeed taken place. We directly quantitated the amount of product in the characteristic band of cycle three, and estimated the amount of product IV present by assuming that the ligation efficiency of the third cycle is the same as that in the first cycle.

Because reactions proceed for the indicated number of cycles, while radiolabeled oligonucleotides are present in the reactions from the start, it is expected that the products of earlier cycles will accumulate as the reaction proceeds. Thus, it is expected and observed that characteristic bands of a particular cycle will be fainter than the bands which account for the product which has been accumulating in the reaction through the previous cycles.

We measured the amount of product in each band on the gel and calculated efficiencies of the latter cycles based on the amount of full-length product produced in the previous cycle. Because the unary counter operates serially through the cycles, we have to consider the amount of product II to be 100% of the template available for the second cycle, and the amount of product III to be 100% of the template available for the third cycle.

For the middle concentration of primers, where the bands were most pronounced and clean, the results showed that 5.3×10^{-3} of template I is converted to product II in cycle one, 1.3×10^{-1} of product II is converted to product III in cycle two, and 3.5×10^{-1} of product III is converted to the characteristic product of cycle three in cycle three. In addition, no product III was generated in the cycle two negative control (NC) reaction in the absence of primer M1 and the presence of M2.

Observed efficiency of the first cycle reactions is fairly low. Consider, however,

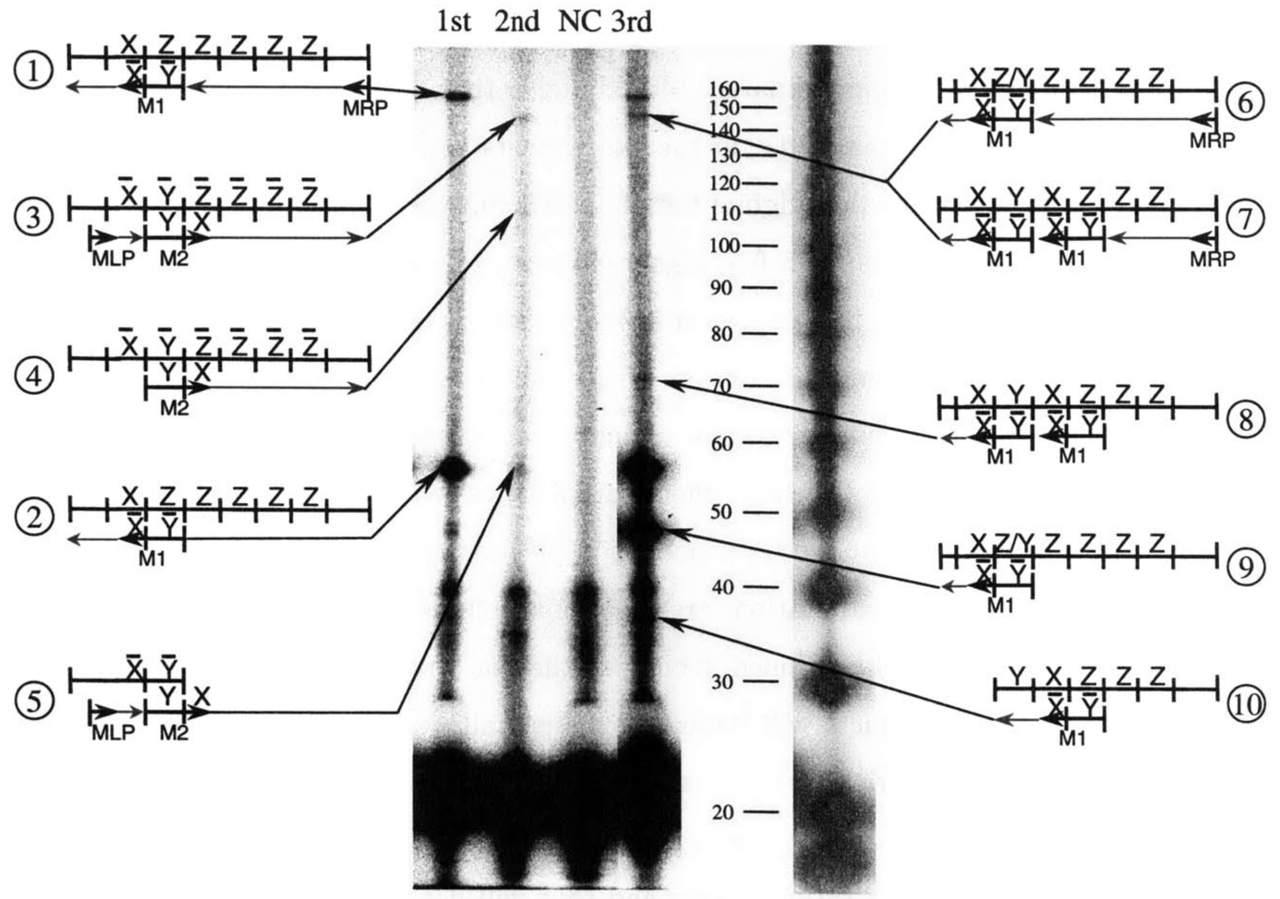


Figure 4-4: Operation of the unary counter machine through cycles 1, 2, and 3, plus negative control (NC) at 0.02 μM of rule oligonucleotides. Negative control reactions contain the same components as second cycle reactions, with the exception of M1, first cycle mutagenic primer. In the absence of M1, no product labeled II in Figure 2-1 is supposed to be produced, and no second cycle product is therefore possible. We expect negative control lanes to contain no product. Oligonucleotide M1 is end-labeled with ^{32}P in Cycles 1 and 3, while M2 is labeled in Cycle 2 and negative control. Cartoons represent the products contained in the corresponding bands. Band labeled 3 is the full-length product of Cycle 2 (and the template for Cycle 3), designated by III in Figure 2-1. Band labeled 8 is the characteristic length product of Cycle 3, produced when the product of extension of MRP fails to ligate to the product of extension of the mutagenic primer M1 annealed in the 3rd cycle location. Other bands as illustrated.

that the first cycle unary counter template is embedded in a 3,000 bases long double stranded DNA vector. Thus, M1 is competing for its binding spot with a 3Kb perfectly complementary strand. In order to produce the full-length product on the first cycle, both MRP and M1 must bind to the same template strand, extend, and ligate together. Furthermore, in the experiment described here, concentration of the original template is approximately 10-fold higher than that of the inner rule oligonucleotides. We have found that in the experiments described in Section 4.3.1, where the concentrations of the original template and rule oligonucleotides were equal, we obtained an approximately 6-fold higher efficiency of the first cycle reaction than observed here. However, it is important to note that those experiments were performed without MLP in the mix and on an earlier generation of the unary counter, which was embedded in a different vector. Thus, detailed comparisons between efficiency data collected in these two experiments are not possible.

We have shown that the basic operations of programmed mutagenesis, which is a universal model of computation, are functional, although the efficiency is low. The major impediment to continued cycling of the machine is that as long as template I is present, its products will increase exponentially with cycle number. However, if the Watson and Crick strands resulting from each DNA replication are separated into different compartments, then the compartment that receives product II will only contain a single computational state and thus will not repeat earlier computational steps. In-vivo programmed mutagenesis might be an effective way to computationally evolve DNA sequences and could potentially assist in sequence specific control of cellular function.

Chapter 5

Conclusions

5.1 Summary of Results

It has been long known that mismatched DNA annealing followed by strand replication can cause the programmed evolution of DNA sequences. We proved that this process is theoretically universal. We showed that programmed mutagenesis is a universal model of computation by directly simulating the smallest known Universal Turing machine.

We demonstrated a constructive way to encode arbitrary computations as DNA molecules, using a single molecule as the memory and state storage and cycles of replication for computational progress. For computer science, using single DNA molecules to represent the state of a computation holds the promise of a new paradigm of composable molecular computing. For biology, the demonstration that DNA sequences could guide their own evolution under computational rules may have implications as we begin to unravel the mysteries of genome encoding and natural evolution.

We have also implemented a unary counter, an example programmed mutagenesis system, and operated it through multiple cycles to collect efficiency data. There is a number of practical limitations apparent in the unary counter system. Chief among these is the limitation imposed by the exponential increase in the amount of the first cycle product. Much attention needs to be devoted to solving this problem, possibly by finding a way to separate the Watson and Crick strands resulting form each DNA replication into separate compartments. While there is no obvious way to perform such separation *in vitro*, perhaps by operating the system *in vivo* separation can be achieved naturally.

5.2 Naturally Occurring Phenomena and Programmed Mutagenesis

It is important to note that the applications of programmed mutagenesis technique do not end with biological computing. In fact, methods we developed would be useful in basic biological research for applications such as creating targeted mutations for basic genetics studies, drug design, gene therapy, in-vitro evolution, and a number of others.

5.2.1 (Deoxy)ribozymes and Programmed Mutagenesis

With the significant progress in creating ribozymes and deoxyribozymes by traditional *in-vitro* evolution methods, many classes and subclasses of ribozymes have been identified. The members of these classes and subclasses share significant structural features, and rational design can and has been used to build new ribozymes with designated features and to combine different features in a single ribozyme, thereby creating novel classes of ribozymes [57]. This process is called rational design. Programmed mutagenesis could be particularly useful in creating new ribozymes and deoxyribozymes by rational design.

Conversely, because ribozymes and deoxyribozymes with many desirable properties, such as self-cleaving [59, 11], self-phosphorylation [38], and self-ligation [58] can now be constructed with virtually an arbitrary sequence in the active site, it is tempting to consider how such tools may be used in programmed mutagenesis systems. There are two interesting applications to be considered.

The first involves the possibility of using deoxyribozyme motif as a readout mechanism. That is, the final rewrite of the computation would create a deoxyribozyme within the template molecule. The deoxyribozyme performing its function would then serve as a readout mechanism.

The second application involves using the product of a deoxyribozyme's function, for example, the piece of DNA spliced off by a self-splicing deoxyribozyme, to initiate the computation. In this case, the newly created fragment of DNA would act as one of the rewrite rules. An additional interesting property of such an application is that by controlling the amount of deoxyribozyme present in the system, we can control the amount of initially available rewrite rule. If that amount is limited, we, in effect, limit the amount of first cycle product produced on the subsequent cycles, thus somewhat limiting the exponential growth of the early cycle product discussed in Chapter 4.

Of course, both of these applications can be used within a single system. A DNA fragment created as a result of the completion of one computation can then be used to initiate another computation in the same test tube, thus creating linked, but strictly sequenced computations.

The RNA world theory has gained much attention in the recent years, especially with the appearance of the many classes of ribozymes produces by *in vitro* evolution. It is possible that if such a world indeed existed, programmed mutagenesis-like events played an active part in evolving RNA enzymes in such an environment.

Other naturally-occurring phenomena, such as RNA editing and gene conversion events could also be possible manifestations of programmed mutagenesis-like processes.

5.2.2 Gene Conversion and Programmed Mutagenesis

Genetic analysis of the products of crosses in haploid organisms, where all four products of a single meiosis can be recovered and examined led to proposals of the models of intragenic recombination, such as gene conversion. A spore pair is produced by mitosis from a single product of meiosis. The mendelian model predicts 4:4 segregation ratios for a cross of two heteroallelic individuals. However aberrant 6:2, 2:6, 5:3, 3:5, and 3:1:1:3 ratios were obtained in such crosses (less than 1% in filamentous fungi, but up to 4% in yeast). These ratios gave the appearance that some alleles in the cross have been "converted" into the other alleles, by the process that was, therefore, named *gene conversion.* In the cases of 6:2 and 2:6 ratios, it appears that the entire chromatid has been converted, while in the cases of the other ratios, it appears that only half-chromatids (one strand of a chromatid) were converted.

There are several models for how gene conversion events occur. The first, and most widely known model is the Holliday model. The key feature of the Holliday model is the formation of the Holliday structure– heteroduplex DNA formed via the hydrogen bonding between two cleaved strands and the complimentary strands in the homologous double helix, followed by ligation. The model further postulates that the cross bridge created as described above then migrates along the two heteroduplex strands, the process termed branch migration. The Holliday structure can then be resolved by cutting and ligating either the two strands that originally participated in an exchange, or the two strands that did not. The model also states that the mismatches in the heteroduplex portion of a molecule can be repaired by an enzymatic system which excises one of the mismatched bases, replacing it with the correct complementary base.

As the data accumulated, it became clear that the Holliday model could not explain the frequencies of occurance of some of the aberrant ratios. It seemed that gene conversion occurred primarily in only one chromatid. The model proposed by Meselson and Radding creates the Holliday structure from a single single-strand cut in one chromatid. The model postulates that following a cut, the $3'$ end of the strand near the cut is extended by the polymerase, using the existing perfectly-matched fragment of the strand as a primer, and displacing the $5'$ fragment of the original strand in the process. Next, the displaced single strand invades the second nonhomologous chromatid duplex, creating a mismatched duplex and forcing the newly displaced strand of the chromatid into a loop, which is then excised. Ligation then forms a Holliday structure, which can be resolved as described above immediately or after branch migration.

The observation that yeast transformation is stimulated 1000-fold if a double-stranded break is introduced into a donor plasmid led to the formulation of a third model, the double-strand-break model, first formulated by Szostack, Orr-Weaver, and Rothstein. Unlike the two models described above, this model uses double-strand breaks to initiate recombination. Digestion of the 5′ ends of both cut sites enlarges the breaks to gaps. The model postulates that the 3′ tail of a strand in a gap invades the uncut duplex of the other chromatid, forcing the displaced strand into a loop, which is then used as a template to fill in the gap in the cut strand which did not participate in the strand displacement. At the same time, the 3' end of the invading strand is extended using the unperturbed strand of the uncut chromatid as a template. Ligations form two Holliday structures, which can be resolved in one of two ways. Finally, mismatch repair is employed to yield gene conversion.

Programmed mutagenesis-like events are postulated in two latter models, which use primer extension and ligation to create DNA structures following a strand break event. The last model is particularly interesting because it postulates polymerization events occurring which use partially mismatched fragments of the DNA strands as primers.

5.2.3 RNA Editing and Programmed Mutagenesis

RNA editing is a process of modification of messenger RNA molecules before translation. Resulting protein sequence is often dramatically different from the gene sequence originally transcribed. RNA editing employs short guide RNA molecules (gRNAs) to mediate the process. These gRNAs are 50-70 nt transcripts which base pair with specific regions of substrate [8] and edits approximately 30 nt of the final transcript [35]. Editing proceeds by insertion, deletion, or substitution of bases exploiting RNA's ability to form G:U base pairs. Each gRNA pairs more efficiently with the final product than with the original substrate. gRNAs contain an A or a G for every inserted U residue.

In the process strikingly similar to programmed mutagenesis, editing proceeds 3′ to 5′ on the mRNA molecule and is dependent on the presence of a full set of overlapping gRNAs. When editing guided by each preceding gRNA finishes, an anchor sequence is created which allows the binding of each successive gRNA, much like incorporation of the rewrite rule M1 allows for binding and incorporation of the rewrite rule M2 (see Figure 2-1) and vice versa.

If we consider gRNAs as rules in the string rewrite system that is RNA editing, we notice that just like the rules in the unary counter, these rules contain read and read/write parts. And just like in the unary counter, processing each preceding rule (i.e. creating a sequence in the mRNA that pairs with every base of this particular gRNA's read/write part) creates an area of the template that becomes the read part for the next rewrite rule gRNA. While the mechanism of editing itself is likely different in programmed mutagenesis and RNA editing, the key feature of sequential incorporation of programmed changes is certainly present in both systems.

5.3 Directions for Future Work

As we noted in Chapter 3, MUTM is not a practical machine to implement in a laboratory because it experiences exponential slowdown. It would be interesting to find another universal Turing machine which fits criteria for encoding established in Section 3.8 and which does not experience an exponential slowdown. If such a machine could be found or designed, developing a programmed mutagenesis encoding for such a machine would be an important research goal. In addition, implementing a small instance of a real computation on such a machine would be a big step forward. Because this would be a universal machine, once it is created, one could run a series of small calculations in the system by changing only the starting template.

More work needs to be done in order to simplify the design process for programmed mutagenesis systems. One of the most limiting factors in the present design of the unary counter is the choice of enzymes and the restrictions it poses on the number and geometry of mismatches allowed. While we were not able to find a better combination of enzymes, it is possible that one can be found, as new and modified thermostable enzymes are introduced every year.

Another possible approach to easing enzyme-posed limitations on the system is to optimize the reaction buffer. While we have found a buffer which allows both enzymes to work efficiently as compared to their native conditions, it may be possible to further optimize the buffer so as to improve the ligation efficiency, and, thus, decrease the cycle time.

Much progress has been made in perfecting our understanding of the influence the mismatches have on the biochemical characteristics of the primers, particularly by the Santa-Lucia laboratory [55]. However, more data on the mismatch biochemistry is needed. Data on the influence of the mismatches on the nearest-neighbor interactions in the DNA duplex is of particular importance. Better data on the mismatch biochemistry would allow us to refine the computer tools and, as a result, to simplify the design process.

We have empirically observed that the geometry of mismatches is the single most important element of the design of the programmed mutagenesis systems. However, at present our understanding of the mechanisms involved is mostly intuitive and anecdotal. We believe a large study of the influence of the geometry of the mismatches on the biochemical characteristics of the DNA duplex is in order. The study should endeavor to elucidate:

1. The precise relationship between the location of the mismatch relative to the site of action of an enzyme and enzyme's efficiency;

2. The comparative degree of instability introduced by the same mismatches depending on their context and distance from each other and the ends of the oligonucleotide; and

3. Whether chemically modifying the oligonucleotides, such as replacing phosphorus linkages by sulfur linkages, changes their biochemical characteristics in general, and in the particular case where some of the mismatches are located in the modified region of the oligonucleotide.

As we observed in Chapter 4, presence in the reaction of the previous cycles' templates as the calculation moves forward is the major impediment to increasing the yields of programmed mutagenesis systems. Investigating *in vitro* and *in vivo* methods for separating products of each cycle of computation is, therefore, an important research goal. In addition to allowing for increased yields of the computation, separating products would also allow for "marking" of compartments. That is, by looking at the product of computation contained in each compartment, one could determine when this compartment was created. If this process occurs *in vivo*, and if the products are separated into different cells, one could determine the lineage of each cell by examining products contained in each cell.

Finally, an interesting research topic would be to design and implement programmed mutagenesis system with deoxyribozyme components. As described in Section 5.2.1, deoxyribozymes can be used as both the readout mechanism and to initiate a computation by providing the first rewrite rule able to act on the template. Next step in this research area would be to put these two approaches together by constructing a system where the output of the first computation would provide an initial rewrite rule for the next computation. This would demonstrate the first experimental instance of a composable biological computing system. In addition, it would be interesting to explore the potential programmed mutagenesis has in assisting in the process of rational design of ribozymes and deoxyribozymes.

To date, programmed mutagenesis is the only composable DNA system to be proven universal. In addition, the basic operations of programmed mutagenesis function, although an improvement in efficiency is highly desirable. Multiple natural phenomena could exhibit programmed mutagenesis-like behavior. We conclude that programmed mutagenesis is an important and promising field in DNA computing, with important theoretical and experimental characteristics deserving of further study.

Appendix A

Encodings and Distance Matrix

Here we provide encodings of the symbols needed to simulate MUTM as well as the distance matrix generated for these encodings.

symbol	encoding
#	CCGGGGGGGGGGC
q_1L	CCAGGGGGGGGGC
q_2L	CCGCGGGGGGGGC
q_2R	CCGGTGGGGGGGC
q_3L	CCGGGAGGGGGGC
q_4L	CCGGGGCGGGGGC
q_5R	CCGGGGGTGGGGC
q_6R	CCGGGGGGAGGGC
q_7L	CCGGGGGGGCGGC
q_7R	CCGGGGGGGGTGC

Table A.1: Encodings of the 13 bp long symbols

symbol	encoding
y	CCATACACAGATAAAGAAATAGACAGGC
0	CCCTCCCCCGCTCACGCACTCGCCCGGC
A	CCTTTCTCTGTTTATGTATTTGTCTGGC
1	CCTATGTGTCTATTTCTTTATCTGTCGC
$q_2/0$	CCATACACAGATAACGCACTCGCCCGGC
y_{2-0}	CCATACACAGATAAAGCACTAGCCAGGC
$q_4/0$	CCCTCCACCGATCAAGAACTAGCCAGGC
y_{4-0}	CCCTCCACCGATAAAGAAATAGACAGGC
$q_7/0$	CCATCCCCCGATCACGAAATAGACCGGC
y_{7-0}	CCATACACAGATCACGAAATAGACCGGC
$q_5/0$	CCCTACCCAGCTAACGCAATCGACAGGC
y_{5-0}	CCATACCCAGCTAAAGAAATCGACAGGC
q_7/y	CCCTCCACAGCTAACGAAATAGCCCGGC
0_{7-y}	CCCTCCACCGCTAACGCACTAGCCCGGC
q_1/y	CCCTACCCAGATCAAGAACTCGACCGGC
0_{1-y}	CCCTACCCAGCTCACGAACTCGCCCGGC
q_2/y	CCATCCACAGCTCAAGCAATCGACCGGC
0_{2-y}	CCATCCCCCGCTCAAGCACTCGACCGGC
q_2/A	CCTTACACTGTTAATGAATTAGTCAGGC
y_{2-A}	CCATACACAGTTAATGAAATAGTCAGGC
q_7/A	CCCTTCCCTGCTTATGTACTTGCCCGGC
0_{7-A}	CCCTCCCCCGCTTATGTACTCGCCCGGC
$q_6/0$	CCTTCCTCCGCTTACGCATTTGCCTGGC
A_{6-0}	CCTTCCTCTGTTTATGCATTTGCCTGGC
$q_2/1$	CCTTTGTCTGTTTATCTTTATCTGTGGC
A_{2-1}	CCTTTCTCTGTTTATCTTTTTCTCTGGC
$q_5/1$	CCTATGTCTCTTTTTGTATATGTGTGGC
A_{5-1}	CCTATCTCTGTTTATGTATATGTGTGGC
$q_6/1$	CCTTTGTGTCTATTTCTATTTGTCTGGC
A_{6-1}	CCTTTCTCTCTATTTGTATTTGTCTGGC
$q_3/1$	CCTATCTGTCTATATGTTTTTCTCTGGC
A_{3-1}	CCTATCTGTGTATATGTATTTGTCTGGC
q_5/A	CCTATGTCTGTATATCTATTTGTGTCGC
1_{5-A}	CCTATGTGTCTATATCTATTTCTGTCGC
q_6/A	CCTTTCTGTCTTTTTGTATATCTCTCGC
1_{6-A}	CCTTTCTGTCTTTTTCTTTATCTGTCGC
q_1/A	CCTATGTCTGTTTTTGTTTTTCTCTCGC
1_{1-A}	CCTATGTCTCTATTTGTTTATCTCTCGC
q_3/A	CCTATCTGTGTTTTTCTTTATGTCTGGC
1_{3-A}	CCTATGTGTGTATTTCTTTATGTGTGGC
q_4/A	CCTTTGTGTCTTTATGTTTTTGTGTCGC
1_{4-A}	CCTTTGTGTCTATATCTTTATGTGTCGC
q_3/y	CCAGACATAAATAGAGACATAGACATGC
y_{3-y}	CCATACACAAATAGAGAAATAGACATGC

Table A.2: Encodings of the 28 bp long symbols

symbol	encoding
q_4/y	CCATAAACATACACAGAAAGAGAAAGGC
y_{4-y}	CCATAAACAGACACAGAAATAGACAGGC
q_5/y	CCACATACAGAGAAATAAATAAATAGGC
y_{5-y}	CCACACACAGATAAAGAAATAAATAGGC
q_6/y	CCATACAAAGATAAAAAGACATACAAGC
y_{6-y}	CCATACAAAGATAAAAAAACAGACAGGC
$q_1/0$	CCCTCACTCGCGCGCGCTCGCACGGC
0_{1-0}	CCCTCCCTCGCTCGCGCGCTCGCCCGGC
$q_1/1$	CCGAGGTGGCTATTGCGTTAGCTGTCGC
1_{1-1}	CCTATGTGGCTATTGCGTTATCTGTCGC
$q_4/1$	CCTATGGGTCGAGTTCTTTAGCGGGCGC
1_{4-1}	CCTATGGGTCGATTTCTTTAGCTGTCGC
$q_7/1$	CCGATGGGGCTAGTTCTTGATCTGGCGC
1_{7-1}	CCGATGTGTCTAGTTCTTGATCTGTCGC
$\sim q_1/y$	CCATACCCAGATAAAGAAGTCGACCGGC
$\sim q_1/0$	CCCTCACCCGCGCACGCTCTCGCACGGC
$\sim q_1/A$	CCTATCTCTGTTTCTGTTTTTGTCTCGC
$\sim q_1/1$	CCGAGGTGTCTATTTCCTTAGCTGTCGC
$\sim q_2/y$	CCATCCACAGCTAAAGCAATGGACAGGC
$\sim q_2/0$	CCATACACGGCTCACGCACTCGCCCGGC
$\sim q_2/A$	CCTTACGCTGTTTATGTATTAGTCAGGC
$\sim q_2/1$	CCTTTGTCTGTCTTTCTTTATCTGTCGC
$\sim q_3/y$	CCAGACATAGATATAGACATAGACAGGC
$\sim q_3/A$	CCTTTCTATGTTTTTGTTTATGTCTGGC
$\sim q_3/1$	CCTATATGTCTATATGTTTATCTCTCGC
$\sim q_4/y$	CCATAGACATATAAAGAAAGAGAAAGGC
$\sim q_4/0$	CCCTCCACCGCTCAGGAACTCGCCAGGC
$\sim q_4/A$	CCTTTGTCTATTTATGTATTTGTGTCGC
$\sim q_4/1$	CCTATGCGTCTAGTTCTTTATCGGGCGC
$\sim q_5/y$	CCATATACAGAGAAATAAATAGAGAGGC
$\sim q_5/0$	CCCTCCCCCGCTAACGCAGTCGACAGGC
$\sim q_5/A$	CCTTTGTCTGTATATCTATTTGTATGGC
$\sim q_5/1$	CCTATGTGTCTATTTATATATGTGTGGC
$\sim q_6/y$	CCATACAGAGATAAAGAGATATACAAGC
$\sim q_6/0$	CCTTCCCCCGCTGACGCATTCGCCTGGC
$\sim q_6/A$	CCTTTCTGTCTTTATGTATTTCTCTAGC
$\sim q_6/1$	CCTTTGTGTCTATTTCTATTTATGTCGC
$\sim q_7/y$	CCATGCACAGCTAACGAAATAGCCAGGC
$\sim q_7/0$	CCATCCCCCGGTCACGAACTCGACCGGC
$\sim q_7/A$	CCCTTCTCTGCTTATGTAGTTGCCTGGC
$\sim q_7/1$	CCCATGGGGCTATTTCTTTATCTGGCGC

Table A.3: Encodings of the 28 bp long symbols (continued)

	y	0	A	1	q_2/0	y_2-0	q_4/0	y_4-0	q_7/0	y_7-0	q_5/0	y_5-0	q_7/y	0_7-y	q_1/y	0_1-y	q_2/y	0_2-y	q_2/A	y_2-A	q_7/A	0_7-A	q_6/0	A_6-0	q_2/1	A_2-1	q_5/1	A_5-1	q_6/1
y	0	12	12	24	6	3	6	3	6	3	6	3	6	9	6	9	6	9	6	3	12	12	12	12	18	15	18	15	18
0	12	0	12	24	6	9	6	9	6	9	6	9	6	3	6	3	6	3	12	12	6	3	6	9	18	15	18	15	18
A	12	12	0	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	6	9	6	9	6	3	6	3	6	3	6
1	24	24	12	0	24	24	24	24	24	24	24	24	24	24	24	24	24	24	18	21	18	21	18	15	6	9	6	9	6
q_2/0	6	6	12	24	0	3	8	9	8	5	6	7	6	5	6	5	6	7	9	7	9	8	9	10	18	15	18	15	18
y_2-0	3	9	12	24	3	0	5	6	9	6	7	6	7	6	7	8	7	8	7	5	10	10	10	10	18	15	18	15	18
q_4/0	6	6	12	24	8	5	0	3	6	7	10	9	6	5	6	7	8	7	8	8	9	7	9	10	18	15	18	15	18
y_4-0	3	9	12	24	9	6	3	0	5	6	7	6	5	6	7	10	7	8	7	6	11	9	10	11	18	15	18	15	18
q_7/0	6	6	12	24	8	9	6	5	0	3	8	7	6	7	6	7	6	5	10	8	10	8	9	11	18	15	18	15	18
y_7-0	3	9	12	24	5	6	7	6	3	0	7	6	5	8	5	6	5	8	8	5	11	11	11	12	18	15	18	15	18
q_5/0	6	6	12	24	6	7	10	7	8	7	0	3	6	7	6	5	6	7	9	7	9	8	9	11	18	15	18	15	18
y_5-0	3	9	12	24	7	6	9	6	7	6	3	0	7	10	5	6	5	6	8	5	10	9	11	12	18	15	18	15	18
q_7/y	6	6	12	24	6	7	6	5	6	5	6	7	0	3	8	5	6	9	8	6	8	7	8	10	18	15	18	15	18
0_7-y	9	3	12	24	5	6	5	6	7	8	7	10	3	0	9	6	7	6	9	9	7	5	6	9	18	15	18	15	18
q_1/y	6	6	12	24	6	7	6	7	6	5	6	5	8	9	0	3	6	5	10	9	8	7	12	12	18	15	18	15	18
0_1-y	9	3	12	24	5	8	7	10	7	6	5	6	5	6	3	0	7	6	10	9	6	5	9	11	18	15	18	15	18
q_2/y	6	6	12	24	6	7	8	7	6	5	6	5	6	7	6	7	0	3	11	8	10	8	9	10	18	15	18	15	18
0_2-y	9	3	12	24	7	8	7	8	5	8	7	6	9	6	5	6	3	0	12	11	8	5	8	10	18	15	18	15	18
q_2/A	6	12	6	18	9	7	8	7	10	8	9	8	8	9	10	10	11	12	0	3	10	11	10	7	12	9	12	9	12
y_2-A	3	12	9	21	7	5	8	6	8	5	7	5	6	9	9	9	8	11	3	0	11	11	12	10	15	12	15	12	15
q_7/A	12	6	6	18	9	10	9	11	10	11	9	10	8	7	8	6	10	8	10	11	0	3	8	7	12	9	12	9	12
0_7-A	12	3	9	21	8	10	7	9	8	11	8	9	7	5	7	5	8	5	11	11	3	0	7	8	15	12	15	12	15
q_6/0	12	6	6	18	9	10	9	10	9	11	9	11	8	6	12	9	9	8	10	12	8	7	0	3	12	9	12	9	12
A_6-0	12	9	3	15	10	10	10	11	11	12	11	12	10	9	12	11	10	10	7	10	7	8	3	0	9	6	9	6	9
q_2/1	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	0	3	6	5	8
A_2-1	15	15	3	9	15	15	15	15	15	15	15	15	15	15	15	15	15	15	9	12	9	12	9	6	3	0	9	6	7
q_5/1	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	6	9	0	3	6
A_5-1	15	15	3	9	15	15	15	15	15	15	15	15	15	15	15	15	15	15	9	12	9	12	9	6	5	6	3	0	9
q_6/1	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	8	7	6	9	0
A_6-1	15	15	3	9	15	15	15	15	15	15	15	15	15	15	15	15	15	15	9	12	9	12	9	6	9	6	5	6	3
q_3/1	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	8	5	8	7	6
A_3-1	15	15	3	9	15	15	15	15	15	15	15	15	15	15	15	15	15	15	9	12	9	12	9	6	9	6	7	4	5
q_5/A	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	6	7	6	5	6
1_5-A	21	21	9	3	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	7	8	7	8	5
q_6/A	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	8	7	6	7	6
1_6-A	21	21	9	3	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	5	6	7	8	7
q_1/A	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	6	5	8	9	6
1_1-A	21	21	9	3	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	7	8	5	8	7
q_3/A	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	6	5	6	5	6
1_3-A	21	21	9	3	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	5	8	5	6	5
q_4/A	18	18	6	6	18	18	18	18	18	18	18	18	18	18	18	18	18	18	12	15	12	15	12	9	6	7	6	7	6
1_4-A	21	21	9	3	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	5	8	7	8	5
q_3/y	6	18	18	24	12	9	12	9	12	9	12	9	12	15	12	15	12	15	12	9	18	18	18	18	23	20	21	20	21
y_3-y	3	15	15	24	9	6	9	6	9	6	9	6	9	12	9	12	9	12	9	6	15	15	15	15	21	18	19	18	19
q_4/y	6	18	18	24	12	9	12	9	12	9	12	9	12	15	12	15	12	15	12	9	18	18	18	18	21	21	19	19	20
y_4-y	3	15	15	24	9	6	9	6	9	6	9	6	9	12	9	12	9	12	9	6	15	15	15	15	20	18	19	18	18
q_5/y	6	18	18	24	12	9	12	9	12	9	12	9	12	15	12	15	12	15	12	9	18	18	18	18	20	19	21	19	21
y_5-y	3	15	15	24	9	6	9	6	9	6	9	6	9	12	9	12	9	12	9	6	15	15	15	15	19	17	19	16	21
q_6/y	6	18	18	24	12	9	12	9	12	9	12	9	12	15	12	15	12	15	12	9	18	18	18	18	20	18	23	20	22
y_6-y	3	15	15	24	9	6	9	6	9	6	9	6	9	12	9	12	9	12	9	6	15	15	15	15	19	17	20	17	19
q_1/0	18	6	18	24	12	15	12	15	12	15	12	15	12	9	12	9	12	9	18	18	12	9	12	15	21	20	21	20	20

	y	0	A	1	q_2/0	y_2-0	q_4/0	y_4-0	q_7/0	y_7-0	q_5/0	y_5-0	q_7/y	0_7-y	q_1/y	0_1-y	q_2/y	0_2-y	q_2/A	y_2-A	q_7/A	0_7-A	q_6/0	A_6-0	q_2/1	A_2-1	q_5/1	A_5-1	q_6/1
0_1-0	15	3	15	24	9	12	9	12	9	12	9	12	9	6	9	6	9	6	15	15	9	6	9	12	20	17	20	18	19
q_1/1	24	24	18	6	24	24	24	24	24	24	24	24	24	24	24	24	24	24	21	22	23	23	20	19	12	15	12	15	12
1_1-1	24	24	15	3	24	24	24	24	24	24	24	24	24	24	24	24	24	24	20	22	21	23	18	17	9	12	9	12	9
q_4/1	24	24	18	6	24	24	24	24	24	24	24	24	24	24	24	24	24	24	20	23	20	22	22	20	12	15	12	15	12
1_4-1	24	24	15	3	24	24	24	24	24	24	24	24	24	24	24	24	24	24	19	22	19	21	20	18	9	12	9	12	9
q_7/1	24	24	18	6	24	24	24	24	24	24	24	24	24	24	24	24	24	24	21	21	20	22	23	21	12	15	12	15	12
1_7-1	24	24	15	3	24	24	24	24	24	24	24	24	24	24	24	24	24	24	20	21	19	22	21	18	9	12	9	12	9
~q_1/y	4	9	12	24	6	5	6	5	7	5	7	6	6	7	3	6	8	8	8	7	9	9	12	12	18	15	18	15	18
~q_1/0	16	4	16	23	10	13	10	13	10	13	10	13	10	7	10	7	10	7	16	16	10	7	10	13	18	17	20	18	20
~q_1/A	16	16	4	9	16	16	16	16	16	16	16	16	16	16	16	16	16	16	10	13	10	13	10	7	7	6	6	7	6
~q_1/1	24	23	16	4	23	23	24	24	24	24	23	24	24	23	24	24	23	23	19	21	21	22	19	16	10	13	10	13	10
~q_2/y	4	9	12	24	7	5	6	5	6	5	7	6	8	9	7	10	3	6	10	7	12	11	10	10	18	15	18	15	18
~q_2/0	9	4	12	24	3	6	6	9	6	6	9	10	7	5	7	6	5	5	11	10	9	7	8	9	18	15	18	15	18
~q_2/A	9	12	4	16	11	9	10	10	11	10	10	10	11	11	11	11	12	12	3	6	8	9	9	6	10	7	10	7	10
~q_2/1	21	21	9	4	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	3	6	6	6	10
~q_3/y	4	16	16	23	10	7	10	7	10	7	10	7	10	13	10	13	10	13	10	7	16	16	16	16	21	18	19	18	19
~q_3/A	16	16	4	9	16	16	16	16	16	16	16	16	16	16	16	16	16	16	10	13	10	13	10	7	6	5	6	5	7
~q_3/1	21	21	9	4	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	6	7	7	6	8
~q_4/y	4	16	16	23	10	7	10	7	10	7	10	7	10	13	10	13	10	13	10	7	16	16	16	16	18	19	17	17	19
~q_4/0	9	4	12	24	8	8	3	6	8	9	8	8	6	5	7	5	7	6	9	9	8	5	8	10	18	15	18	15	18
~q_4/A	16	16	4	9	16	16	16	16	16	16	16	16	16	16	16	16	16	16	10	13	10	13	10	7	6	5	7	5	9
~q_4/1	24	23	16	4	24	24	24	24	23	24	23	23	24	24	23	23	24	23	19	22	18	21	21	18	10	13	10	13	10
~q_5/y	4	16	16	23	10	7	10	7	10	7	10	7	10	13	10	13	10	13	10	7	16	16	16	16	18	18	19	17	19
~q_5/0	9	4	12	24	8	9	8	6	7	10	3	6	7	5	8	7	7	5	10	10	9	6	7	10	18	15	18	15	18
~q_5/A	16	16	4	9	16	16	16	16	16	16	16	16	16	16	16	16	16	16	10	13	10	13	10	7	6	5	8	5	6
~q_5/1	21	21	9	4	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	8	11	3	6	6
~q_6/y	4	16	16	23	10	7	10	7	10	7	10	7	10	13	10	13	10	13	10	7	16	16	16	16	20	17	22	19	20
~q_6/0	12	4	9	21	8	10	9	10	8	11	7	9	8	6	10	7	8	6	10	12	9	6	3	6	15	12	15	12	15
~q_6/A	16	16	4	9	16	16	16	16	16	16	16	16	16	16	16	16	16	16	10	13	10	13	10	7	8	5	8	7	6
~q_6/1	21	21	9	4	21	21	21	21	21	21	21	21	21	21	21	21	21	21	15	18	15	18	15	12	7	8	6	9	3
~q_7/y	4	9	12	24	6	5	7	6	7	5	8	5	3	6	8	7	5	8	8	5	9	9	10	11	18	15	18	15	18
~q_7/0	9	4	12	24	7	8	5	8	3	6	10	9	6	5	7	5	8	5	10	9	8	6	8	10	18	15	18	15	18
~q_7/A	12	9	4	16	11	11	10	11	12	12	10	11	9	9	11	9	11	11	10	11	3	6	6	5	10	7	10	7	10
~q_7/1	24	23	16	4	24	24	23	23	24	24	23	24	23	23	23	23	24	24	20	21	18	20	21	19	10	13	10	13	10

	A_6-1	q_3/1	A_3-1	q_5/A	1_5-A	q_6/A	1_6-A	q_1/A	1_1-A	q_3/A	1_3-A	q_4/A	1_4-A	q_3/y	y_3-y	q_4/y	y_4-y	q_5/y	y_5-y	q_6/y	y_6-y	q_1/0	0_1-0	q_1/1	1_1-1	q_4/1	1_4-1	q_7/1	1_7-1
y	15	18	15	18	21	18	21	18	21	18	21	18	21	6	3	6	3	6	3	6	3	18	15	24	24	24	24	24	24
0	15	18	15	18	21	18	21	18	21	18	21	18	21	18	15	18	15	18	15	18	15	6	3	24	24	24	24	24	24
A	3	6	3	6	9	6	9	6	9	6	9	6	9	18	15	18	15	18	15	18	15	18	15	18	15	18	15	18	15
1	9	6	9	6	3	6	3	6	3	6	3	6	3	24	24	24	24	24	24	24	24	24	24	6	3	6	3	6	3
q_2/0	15	18	15	18	21	18	21	18	21	18	21	18	21	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24
y_2-0	15	18	15	18	21	18	21	18	21	18	21	18	21	9	6	9	6	9	6	9	6	15	12	24	24	24	24	24	24
q_4/0	15	18	15	18	21	18	21	18	21	18	21	18	21	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24
y_4-0	15	18	15	18	21	18	21	18	21	18	21	18	21	9	6	9	6	9	6	9	6	15	12	24	24	24	24	24	24
q_7/0	15	18	15	18	21	18	21	18	21	18	21	18	21	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24
y_7-0	15	18	15	18	21	18	21	18	21	18	21	18	21	9	6	9	6	9	6	9	6	15	12	24	24	24	24	24	24
q_5/0	15	18	15	18	21	18	21	18	21	18	21	18	21	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24
y_5-0	15	18	15	18	21	18	21	18	21	18	21	18	21	9	6	9	6	9	6	9	6	15	12	24	24	24	24	24	24
q_7/y	15	18	15	18	21	18	21	18	21	18	21	18	21	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24
0_7-y	15	18	15	18	21	18	21	18	21	18	21	18	21	15	12	15	12	15	12	15	12	9	6	24	24	24	24	24	24
q_1/y	15	18	15	18	21	18	21	18	21	18	21	18	21	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24
0_1-y	15	18	15	18	21	18	21	18	21	18	21	18	21	15	12	15	12	15	12	15	12	9	6	24	24	24	24	24	24
q_2/y	15	18	15	18	21	18	21	18	21	18	21	18	21	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24
0_2-y	15	18	15	18	21	18	21	18	21	18	21	18	21	15	12	15	12	15	12	15	12	9	6	24	24	24	24	24	24
q_2/A	9	12	9	12	15	12	15	12	15	12	15	12	15	12	9	12	9	12	9	12	9	18	15	21	20	20	19	21	20
y_2-A	12	15	12	15	18	15	18	15	18	15	18	15	18	9	6	9	6	9	6	9	6	18	15	22	22	23	22	21	21
q_7/A	9	12	9	12	15	12	15	12	15	12	15	12	15	18	15	18	15	18	15	18	15	12	9	23	21	20	19	20	19
0_7-A	12	15	12	15	18	15	18	15	18	15	18	15	18	18	15	18	15	18	15	18	15	9	6	23	23	22	21	22	22
q_6/0	9	12	9	12	15	12	15	12	15	12	15	12	15	18	15	18	15	18	15	18	15	12	9	20	18	22	20	23	21
A_6-0	6	9	6	9	12	9	12	9	12	9	12	9	12	18	15	18	15	18	15	18	15	15	12	19	17	20	18	21	18
q_2/1	9	8	9	6	7	8	5	6	7	6	5	6	5	23	21	21	20	20	19	20	19	21	20	12	9	12	9	12	9
A_2-1	6	5	6	7	8	7	6	5	8	5	8	7	8	20	18	21	18	19	17	18	17	20	17	15	12	15	12	15	12
q_5/1	5	8	7	6	7	6	7	8	5	6	5	6	7	21	19	19	19	21	19	23	20	21	20	12	9	12	9	12	9
A_5-1	6	7	4	5	8	7	8	9	8	5	6	7	8	20	18	19	18	19	16	20	17	20	18	15	12	15	12	15	12
q_6/1	3	6	5	6	5	6	7	6	7	6	5	6	5	21	19	20	18	21	21	22	19	20	19	12	9	12	9	12	9
A_6-1	0	5	4	7	8	5	8	5	6	7	8	7	8	19	16	18	16	20	18	21	18	19	17	15	12	15	12	15	12
q_3/1	5	0	3	8	5	6	7	6	5	6	7	6	7	20	20	22	20	21	19	21	20	21	19	12	9	12	9	12	9
A_3-1	4	3	0	5	6	7	10	7	8	5	6	7	8	19	18	20	17	19	17	20	17	19	17	15	12	15	12	15	12
q_5/A	7	8	5	0	3	10	9	6	7	8	5	6	5	22	20	21	19	19	19	22	20	21	21	12	9	12	9	12	9
1_5-A	8	5	6	3	0	7	6	7	6	9	6	5	4	23	22	23	22	21	21	23	22	23	23	9	6	9	6	9	6
q_6/A	5	6	7	10	7	0	3	6	5	6	9	6	7	20	18	21	20	23	20	20	19	22	19	12	9	12	9	12	9
1_6-A	8	7	10	9	6	3	0	7	6	5	6	5	4	22	21	23	23	24	22	21	21	23	21	9	6	9	6	9	6
q_1/A	5	6	7	6	7	6	7	0	3	8	7	6	7	21	19	21	18	21	20	21	21	20	19	12	9	12	9	12	9
1_1-A	6	5	8	7	6	5	6	3	0	7	6	7	6	22	21	22	21	23	22	23	23	23	22	9	6	9	6	9	6
q_3/A	7	6	5	8	9	6	5	8	7	0	3	8	7	20	20	22	20	22	20	20	18	21	18	12	9	12	9	12	9
1_3-A	8	7	6	5	6	9	6	7	6	3	0	7	4	23	23	22	21	22	22	23	21	21	21	9	6	9	6	9	6
q_4/A	7	6	7	6	5	6	5	6	7	8	7	0	3	20	19	21	20	22	20	21	20	20	19	12	9	12	9	12	9
1_4-A	8	7	8	5	4	7	4	7	6	7	4	3	0	23	22	22	22	23	23	22	21	22	22	9	6	9	6	9	6
q_3/y	19	20	19	22	23	20	22	21	22	20	23	20	23	0	3	10	8	11	8	9	8	19	16	24	24	24	24	24	24
y_3-y	16	20	18	20	22	18	21	19	21	20	23	19	22	3	0	7	5	9	6	8	6	19	16	24	24	24	24	24	24
q_4/y	18	22	20	21	23	21	23	21	22	22	22	21	22	10	7	0	3	9	8	11	8	18	20	24	24	24	24	24	24
y_4-y	16	20	17	19	22	20	23	18	21	20	21	20	22	8	5	3	0	7	6	9	6	17	17	24	24	24	24	24	24
q_5/y	20	21	19	19	21	23	24	21	23	22	22	22	23	11	9	9	7	0	3	10	8	20	21	24	24	24	24	24	24
y_5-y	18	19	17	19	21	20	22	20	22	20	22	20	23	8	6	8	6	3	0	8	6	20	18	24	24	24	24	24	24
q_6/y	21	21	20	22	23	20	21	21	23	20	23	21	22	9	8	11	9	10	8	0	3	21	18	24	24	24	24	24	24
y_6-y	18	20	17	20	22	19	21	21	23	18	21	20	21	8	6	8	6	8	6	3	0	20	17	24	24	24	24	24	24
q_1/0	19	21	19	21	23	22	23	20	23	21	21	20	22	19	19	18	17	20	20	21	20	0	3	24	24	24	24	24	24

	A_6-1	q_3/1	A_3-1	q_5/A	1_5-A	q_6/A	1_6-A	q_1/A	1_1-A	q_3/A	1_3-A	q_4/A	1_4-A	q_3/y	y_3-y	q_4/y	y_4-y	q_5/y	y_5-y	q_6/y	y_6-y	q_1/0	0_1-0	q_1/1	1_1-1	q_4/1	1_4-1	q_7/1	1_7-1
0_1-0	17	19	17	21	23	19	21	19	22	18	21	19	22	16	16	20	17	21	18	18	17	3	0	24	24	24	24	24	24
q_1/1	15	12	15	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24	24	24	24	24	0	3	10	7	8	7
1_1-1	12	9	12	9	6	9	6	9	6	9	6	9	6	24	24	24	24	24	24	24	24	24	24	3	0	9	6	7	6
q_4/1	15	12	15	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24	24	24	24	24	10	9	0	3	6	7
1_4-1	12	9	12	9	6	9	6	9	6	9	6	9	6	24	24	24	24	24	24	24	24	24	24	7	6	3	0	7	6
q_7/1	15	12	15	12	9	12	9	12	9	12	9	12	9	24	24	24	24	24	24	24	24	24	24	8	7	6	7	0	3
1_7-1	12	9	12	9	6	9	6	9	6	9	6	9	6	24	24	24	24	24	24	24	24	24	24	7	6	7	6	3	0
~q_1/y	15	18	15	18	21	18	21	18	21	18	21	18	21	10	7	10	7	10	7	10	7	15	12	24	24	23	23	23	24
~q_1/0	18	19	18	19	22	22	22	18	21	20	20	18	20	21	19	17	16	18	18	21	19	3	6	23	23	23	23	23	23
~q_1/A	5	9	7	6	9	6	8	3	6	7	8	5	8	18	17	20	17	21	19	20	19	19	17	15	12	15	12	15	12
~q_1/1	13	10	13	10	7	10	7	10	7	10	7	10	7	24	24	24	24	24	24	24	24	23	23	3	6	8	5	8	5
~q_2/y	15	18	15	18	21	18	21	18	21	18	21	18	21	10	7	10	7	10	7	10	7	15	12	23	24	23	23	24	24
~q_2/0	15	18	15	18	21	18	21	18	21	18	21	18	21	15	12	15	12	15	12	15	12	10	7	23	23	24	24	23	24
~q_2/A	7	10	7	10	13	10	13	10	13	10	13	10	13	15	12	15	12	15	12	15	12	18	15	20	19	18	16	19	19
~q_2/1	11	9	10	5	6	8	5	6	5	6	5	7	6	23	22	22	21	22	21	22	22	23	22	10	7	10	7	10	7
~q_3/y	17	19	17	21	23	19	21	19	21	17	20	20	23	3	6	9	6	9	6	8	6	18	15	23	23	23	23	23	23
~q_3/A	5	7	6	10	12	5	6	6	7	3	6	7	8	19	18	20	18	22	19	18	16	19	16	15	12	15	12	15	12
~q_3/1	8	3	6	8	5	7	5	9	6	6	5	7	5	23	23	23	23	21	21	22	21	23	22	10	7	10	7	10	7
~q_4/y	18	21	19	19	21	20	22	20	21	21	21	18	20	9	6	3	6	8	6	9	6	19	19	23	23	23	23	23	23
~q_4/0	15	18	15	18	21	18	21	18	21	18	21	18	21	15	12	15	12	15	12	15	12	10	7	23	23	24	24	24	24
~q_4/A	6	7	7	6	8	7	6	6	8	8	9	3	6	18	16	20	19	21	18	20	19	20	18	15	12	15	12	15	12
~q_4/1	13	10	13	10	7	10	7	10	7	10	7	10	7	24	24	24	24	24	24	24	24	23	23	10	7	3	6	5	5
~q_5/y	18	21	18	17	20	22	22	20	23	21	20	19	20	10	7	7	5	3	6	9	6	18	19	23	23	23	23	23	23
~q_5/0	15	18	15	18	21	18	21	18	21	18	21	18	21	15	12	15	12	15	12	15	12	10	7	24	24	24	24	23	23
~q_5/A	5	8	5	3	6	9	8	7	10	8	7	7	6	20	17	20	18	19	18	20	18	20	19	15	12	15	12	15	12
~q_5/1	8	9	8	6	5	5	5	9	6	6	5	5	5	22	21	22	22	23	22	22	20	23	22	10	7	10	7	10	7
~q_6/y	19	18	17	21	21	18	20	19	22	19	22	18	21	7	6	10	7	9	6	3	6	19	16	23	23	23	23	23	23
~q_6/0	12	15	12	15	18	15	18	15	18	15	18	15	18	18	15	18	15	18	15	18	15	10	7	22	21	21	21	23	22
~q_6/A	5	6	5	9	8	3	6	5	8	6	9	7	10	19	17	21	18	21	18	18	18	20	17	15	12	15	12	15	12
~q_6/1	6	6	7	8	5	5	6	7	5	6	5	9	6	23	22	22	21	21	21	23	21	23	22	10	7	10	7	10	7
~q_7/y	15	18	15	18	21	18	21	18	21	18	21	18	21	10	7	10	7	10	7	10	7	15	12	23	24	24	24	24	24
~q_7/0	15	18	15	18	21	18	21	18	21	18	21	18	21	15	12	15	12	15	12	15	12	10	7	24	24	23	23	24	24
~q_7/A	7	10	7	10	13	10	13	10	13	10	13	10	13	18	15	18	15	18	15	18	15	15	12	21	19	20	18	19	16
~q_7/1	13	10	13	10	7	10	7	10	7	10	7	10	7	24	24	24	24	24	24	24	24	23	23	7	5	6	5	3	6

	~q_1/y	~q_1/0	~q_1/A	~q_1/1	~q_2/y	~q_2/0	~q_2/A	~q_2/1	~q_3/y	~q_3/A	~q_3/1	~q_4/y	~q_4/0	~q_4/A	~q_4/1	~q_5/y	~q_5/0	~q_5/A	~q_5/1	~q_6/y	~q_6/0	~q_6/A	~q_6/1	~q_7/y	~q_7/0	~q_7/A	~q_7/1
y	4	16	16	24	4	9	9	21	4	16	21	4	9	16	24	4	9	16	21	4	12	16	21	4	9	12	24
0	9	4	16	23	9	4	12	21	16	16	21	16	4	16	23	16	4	16	21	16	4	16	21	9	4	9	23
A	12	16	4	16	12	12	4	9	16	4	9	16	12	4	16	16	12	4	9	16	9	4	9	12	12	4	16
1	24	23	9	4	24	24	16	4	23	9	4	23	24	9	4	23	24	9	4	23	21	9	4	24	24	16	4
q_2/0	6	10	16	23	7	3	11	21	10	16	21	10	8	16	24	10	8	16	21	10	8	16	21	6	7	11	24
y_2-0	5	13	16	23	5	6	9	21	7	16	21	7	8	16	24	7	9	16	21	7	10	16	21	5	8	11	24
q_4/0	6	10	16	24	6	6	10	21	10	16	21	10	3	16	24	10	8	16	21	10	9	16	21	7	5	10	23
y_4-0	5	13	16	24	5	9	10	21	7	16	21	7	6	16	24	7	6	16	21	7	10	16	21	6	8	11	23
q_7/0	7	10	16	24	6	6	11	21	10	16	21	10	8	16	23	10	7	16	21	10	8	16	21	7	3	12	24
y_7-0	5	13	16	24	5	6	10	21	7	16	21	7	9	16	24	7	10	16	21	7	11	16	21	5	6	12	24
q_5/0	7	10	16	23	7	9	10	21	10	16	21	10	8	16	23	10	3	16	21	10	7	16	21	8	10	10	23
y_5-0	6	13	16	24	6	10	10	21	7	16	21	7	8	16	23	7	6	16	21	7	9	16	21	5	9	11	24
q_7/y	6	10	16	24	8	7	11	21	10	16	21	10	6	16	24	10	7	16	21	10	8	16	21	3	6	9	23
0_7-y	7	7	16	23	9	5	11	21	13	16	21	13	5	16	24	13	5	16	21	13	6	16	21	6	5	9	23
q_1/y	3	10	16	24	7	7	11	21	10	16	21	10	7	16	23	10	8	16	21	10	10	16	21	8	7	11	23
0_1-y	6	7	16	24	10	6	11	21	13	16	21	13	5	16	23	13	7	16	21	13	7	16	21	7	5	9	23
q_2/y	8	10	16	23	3	5	12	21	10	16	21	10	7	16	24	10	7	16	21	10	8	16	21	5	8	11	24
0_2-y	8	7	16	23	6	5	12	21	13	16	21	13	6	16	23	13	5	16	21	13	6	16	21	8	5	11	24
q_2/A	8	16	10	19	10	11	3	15	10	10	15	10	9	10	19	10	10	10	15	10	10	10	15	8	10	10	20
y_2-A	7	16	13	21	7	10	6	18	7	13	18	7	9	13	22	7	10	13	18	7	12	13	18	5	9	11	21
q_7/A	9	10	10	21	12	9	8	15	16	10	15	16	8	10	18	16	9	10	15	16	9	10	15	9	8	3	18
0_7-A	9	7	13	22	11	7	9	18	16	13	18	16	5	13	21	16	6	13	18	16	6	13	18	9	6	6	20
q_6/0	12	10	10	19	10	8	9	15	16	10	15	16	8	10	21	16	7	10	15	16	3	10	15	10	8	6	21
A_6-0	12	13	7	16	10	9	6	12	16	7	12	16	10	7	18	16	10	7	12	16	6	7	12	11	10	5	19
q_2/1	18	18	7	10	18	18	10	3	21	6	6	18	18	6	10	18	18	6	8	20	15	8	7	18	18	10	10
A_2-1	15	17	6	13	15	15	7	6	18	5	7	19	15	5	13	18	15	5	11	17	12	5	8	15	15	7	13
q_5/1	18	20	6	10	18	18	10	6	19	6	7	17	18	7	10	19	18	8	3	22	15	8	6	18	18	10	10
A_5-1	15	18	7	13	15	15	7	6	18	5	6	17	15	5	13	17	15	5	6	19	12	7	9	15	15	7	13
q_6/1	18	20	6	10	18	18	10	10	19	7	8	19	18	9	10	19	18	6	6	20	15	6	3	18	18	10	10
A_6-1	15	18	5	13	15	15	7	11	17	5	8	18	15	6	13	18	15	5	8	19	12	5	6	15	15	7	13
q_3/1	18	19	9	10	18	18	10	9	19	7	3	21	18	7	10	21	18	8	9	18	15	6	6	18	18	10	10
A_3-1	15	18	7	13	15	15	7	10	17	6	6	19	15	7	13	18	15	5	8	17	12	5	7	15	15	7	13
q_5/A	18	19	6	10	18	18	10	5	21	10	8	19	18	6	10	17	18	3	6	21	15	9	8	18	18	10	10
1_5-A	21	22	9	7	21	21	13	6	23	12	5	21	21	8	7	20	21	6	5	21	18	8	5	21	21	13	7
q_6/A	18	22	6	10	18	18	10	8	19	5	7	20	18	7	10	22	18	9	5	18	15	3	5	18	18	10	10
1_6-A	21	22	8	7	21	21	13	5	21	6	5	22	21	6	7	22	21	8	5	20	18	6	6	21	21	13	7
q_1/A	18	18	3	10	18	18	10	6	19	6	9	20	18	6	10	20	18	7	9	19	15	5	7	18	18	10	10
1_1-A	21	21	6	7	21	21	13	5	21	7	6	21	21	8	7	23	21	10	6	22	18	8	5	21	21	13	7
q_3/A	18	20	7	10	18	18	10	6	17	3	6	21	18	8	10	21	18	8	6	19	15	6	6	18	18	10	10
1_3-A	21	20	8	7	21	21	13	5	20	6	5	21	21	9	7	20	21	7	5	22	18	9	5	21	21	13	7
q_4/A	18	18	5	10	18	18	10	7	20	7	7	18	18	3	10	19	18	7	5	18	15	7	9	18	18	10	10
1_4-A	21	20	8	7	21	21	13	6	23	8	5	20	21	6	7	20	21	6	5	21	18	10	6	21	21	13	7
q_3/y	10	21	18	24	10	15	15	23	3	19	23	9	15	18	24	10	15	20	22	7	18	19	23	10	15	18	24
y_3-y	7	19	17	24	7	12	12	22	6	18	23	6	12	16	24	7	12	17	21	6	15	17	22	7	12	15	24
q_4/y	10	17	20	24	10	15	15	22	9	20	23	3	15	20	24	7	15	20	22	10	18	21	22	10	15	18	24
y_4-y	7	16	17	24	7	12	12	21	6	18	23	6	12	19	24	5	12	18	22	7	15	18	21	7	12	15	24
q_5/y	10	18	21	24	10	15	15	22	9	22	21	8	15	21	24	3	15	19	23	9	18	21	21	10	15	18	24
y_5-y	7	18	19	24	7	12	12	21	6	19	21	6	12	18	24	6	12	18	22	6	15	18	21	7	12	15	24
q_6/y	10	21	20	24	10	15	15	22	8	18	22	9	15	20	24	9	15	20	22	3	18	18	23	10	15	18	24
y_6-y	7	19	19	24	7	12	12	22	6	16	21	6	12	19	24	6	12	18	20	6	15	18	21	7	12	15	24
q_1/0	15	3	19	23	15	10	18	23	18	19	23	19	10	20	23	18	10	20	23	19	10	20	23	15	10	15	23

	~q_1/y	~q_1/0	~q_1/A	~q_1/1	~q_2/y	~q_2/0	~q_2/A	~q_2/1	~q_3/y	~q_3/A	~q_3/1	~q_4/y	~q_4/0	~q_4/A	~q_4/1	~q_5/y	~q_5/0	~q_5/A	~q_5/1	~q_6/y	~q_6/0	~q_6/A	~q_6/1	~q_7/y	~q_7/0	~q_7/A	~q_7/1
0_1-0	12	6	17	23	12	7	15	22	15	16	22	19	7	18	23	19	7	19	22	16	7	17	22	12	7	12	23
q_1/1	24	23	15	3	23	23	20	10	23	15	10	23	23	15	10	23	24	15	10	23	22	15	10	23	24	21	7
1_1-1	24	23	12	6	24	23	19	7	23	12	7	23	23	12	7	23	24	12	7	23	21	12	7	24	24	19	5
q_4/1	23	23	15	8	23	24	18	10	23	15	10	23	24	15	3	23	24	15	10	23	21	15	10	24	23	20	6
1_4-1	23	23	12	5	23	24	16	7	23	12	7	23	24	12	6	23	24	12	7	23	21	12	7	24	23	18	5
q_7/1	23	23	15	8	24	23	19	10	23	15	10	23	24	15	5	23	23	15	10	23	23	15	10	24	24	19	3
1_7-1	24	23	12	5	24	24	19	7	23	12	7	23	24	12	5	23	23	12	7	23	22	12	7	24	24	16	6
~q_1/y	0	13	16	24	8	9	9	21	8	16	21	8	9	16	24	8	9	16	21	8	12	16	21	6	8	11	22
~q_1/0	13	0	18	22	13	8	16	21	19	18	21	17	8	17	22	16	8	18	23	19	8	20	23	13	8	13	22
~q_1/A	16	18	0	13	16	16	8	7	16	5	12	18	16	5	13	19	16	7	7	18	13	5	9	16	16	8	13
~q_1/1	24	22	13	0	22	23	18	8	23	13	8	23	24	13	8	23	23	13	8	23	21	13	8	23	24	19	7
~q_2/y	8	13	16	22	0	6	11	21	8	16	21	8	8	16	24	8	8	16	21	8	10	16	21	7	9	12	24
~q_2/0	9	8	16	23	6	0	12	21	13	16	21	13	6	16	24	13	8	16	21	13	7	16	21	8	5	11	23
~q_2/A	9	16	8	18	11	12	0	13	13	8	13	13	11	8	18	13	11	8	13	13	10	8	13	11	11	8	17
~q_2/1	21	21	7	8	21	21	13	0	22	8	7	21	21	7	8	21	21	8	6	22	18	9	8	21	21	13	8
~q_3/y	8	19	16	23	8	13	13	22	0	16	22	8	13	19	23	8	13	20	21	6	16	17	21	8	13	16	23
~q_3/A	16	18	5	13	16	16	8	8	16	0	8	19	16	6	13	20	16	8	8	18	13	5	8	16	16	8	13
~q_3/1	21	21	12	8	21	21	13	7	22	8	0	22	21	8	8	20	21	8	7	21	18	9	6	21	21	13	8
~q_4/y	8	17	18	23	8	13	13	21	8	19	22	0	13	18	23	6	13	19	20	8	16	20	21	8	13	16	23
~q_4/0	9	8	16	24	8	6	11	21	13	16	21	13	0	16	24	13	6	16	21	13	7	16	21	8	6	9	23
~q_4/A	16	17	5	13	16	16	8	7	19	6	8	18	16	0	13	18	16	5	8	18	13	7	12	16	16	8	13
~q_4/1	24	22	13	8	24	24	18	8	23	13	8	23	24	13	0	23	23	13	8	23	20	13	8	24	23	19	5
~q_5/y	8	16	19	23	8	13	13	21	8	20	20	6	13	18	23	0	13	16	21	8	16	20	22	8	13	16	23
~q_5/0	9	8	16	23	8	8	11	21	13	16	21	13	6	16	23	13	0	16	21	13	5	16	21	10	8	9	23
~q_5/A	16	18	7	13	16	16	8	8	20	8	8	19	16	5	13	16	16	0	9	19	13	7	9	16	16	8	13
~q_5/1	21	23	7	8	21	21	13	6	21	8	7	20	21	8	8	21	21	9	0	22	18	8	6	21	21	13	8
~q_6/y	8	19	18	23	8	13	13	22	6	18	21	8	13	18	23	8	13	19	22	0	16	15	22	8	13	16	23
~q_6/0	12	8	13	21	10	7	10	18	16	13	18	16	7	13	20	16	5	13	18	16	0	13	18	10	7	9	23
~q_6/A	16	20	5	13	16	16	8	9	17	5	9	20	16	7	13	20	16	7	8	15	13	0	7	16	16	8	13
~q_6/1	21	23	9	8	21	21	13	8	21	8	6	21	21	12	8	22	21	9	6	22	18	7	0	21	21	13	8
~q_7/y	6	13	16	23	7	8	11	21	8	16	21	8	8	16	24	8	10	16	21	8	10	16	21	0	7	10	24
~q_7/0	8	8	16	24	9	5	11	21	13	16	21	13	6	16	23	13	8	16	21	13	7	16	21	7	0	11	24
~q_7/A	11	13	8	19	12	11	8	13	16	8	13	16	9	8	19	16	9	8	13	16	9	8	13	10	11	0	18
~q_7/1	22	22	13	7	24	23	17	8	23	13	8	23	23	13	5	23	23	13	8	23	23	13	8	24	24	18	0

Bibliography

[1] F. Aboul-ela, D. Koh, and I. Tinoco, Jr. Base-base mismatches. Thermodynamics of double helix formation for dCA3XA3G + dCT3YT3G (X, Y = A, C, G, T). *Nucleic Acids Research*, 13(13):4813–4823, 1985.

[2] L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994.

[3] L.M. Adleman. On constructing a molecular computer. Technical report, UCLA, Department of Computer Science, UCLA, CA, January 1995.

[4] I. Ausubel and M. Frederick (eds.). *Current Protocols in Molecular Biology*, chapter 8.5. John Wiley & Sons, Inc., 1997.

[5] D. Beaver. Molecular computing. Technical report, Department of Computer Science and Engineering Technical Report, Penn State University, 1995.

[6] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414(6862):430–434, 2001.

[7] New England Biolabs. *Biological research products catalog.* NEBiolabs, Inc., 1997.

[8] B. Blum, N. Bakalara, and L. Simpson. A model for RNA editing in kinetoplasmid mitochondria: "guide" RNA molecules transcribed from maxicircle DNA provide the edited information. *Cell*, 60:189–198, 1990.

[9] R. S. Braich, C. Johnson, P. W. K. Rothemund, D. Hwang, N. Chelyapov, and L. M. Adleman. Solution of a satisfability problem on a gel-based DNA computer. In *Lecture Notes in Computer Science, 2054, DNA Computing, A. Condon, G. Rozenberg (eds.)*, pages 27–42, 2001.

[10] K.J. Breslauer, H. Blocker, R. Frank, and L.A. Marky. Predicting DNA duplex stability from the base sequence. *PNAS USA*, 83(11):3746–3750, 1986.

[11] N. Carmi, S.R. Balkhi, and R.R. Breaker. Cleaving DNA with DNA. *PNAS USA*, 95:2233–2237, 1998.

[12] Erzsébet Csuhaj-Varjú, Rudolf Freund, Lila Kari, and Gheorghe Păun. DNA computing based on splicing: universality results. In *Biocomputing: Proceedings of the 1996 Pacific Symposium*, pages 179–190, 1996.

[13] PI D. K. Gifford. Programmed mutagenesis and receptor-based sensor cells. *Proposal to DARPA ITO BAA#98-11*, 1997.

[14] M.B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[15] Tony L. Eng. Linear DNA self-assembly with hairpins generates the equivalent of linear context-free grammars. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 289–296, 1997.

[16] Tony L. Eng and Benjamin M. Serridge. A surface-based DNA algorithm for minimal set cover. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 185–192, 1999.

[17] Rudolf Freund, Erzsébet Csuhaj-Varjú, and Franz Wachtler. Test tube systems with cutting/recombination operations. In *Pasific Symposium on Biocomputing, R.B. Altman, A.K. Dunker, L. Hunter, T.E. Klein (eds.)*, pages 163–174, 1997.

[18] Anthony G. Frutos, Andrew J. Thiel, Anne E. Condon, Lloyd M. Smith, and Robert M. Corn. DNA computing at surfaces: 4 base mismatch word design. In *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23 - 25, 1997*, page 238, 1997.

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

[20] M. Hagiya, M. Arita, D. Kiga, K. Sakomoto, and S. Yokoyama. Towards parallel evaluation and learning of boolean mu-formulas with molecules. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 57–72, 1999.

[21] A.J. Hartemink and D.K. Gifford. Thermodynamic simulation of deoxyoligonucleotide hybridization for DNA computation. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 25–38, 1999.

[22] A.J. Hartemink, D.K. Gifford, and J. Khodor. Automated constraint-governed nucleotide sequence selection for DNA computation. *BioSystems*, 52:227–235, 1998.

[23] Thomas Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49:737–759, 1987.

[24] Nataša Jonoska, Stephen A. Karl, and Masahico Saito. Creating 3-dimensional graph structures with DNA. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 123–136, 1999.

[25] Natasa Jonoska and Masahico Saito. Boundary compunents of thickened graphs. In *Proceedings 7th DIMACS Workshop on DNA Based Computers, Tampa, FL, USA, June 2001*, pages 148–158, 2001.

[26] Peter D. Kaplan, Guillermo Cecchi, and Albert Libchaber. DNA based molecular computation: template-template interactions in PCR. In *Proceedings of the Second Annual Meeting on DNA Based Computers, held at Princeton University, June 10-12, 1996*, 1996.

[27] L. Kari and G. Thierin. Contextual insertions/deletions and computability. *Information and Computation*, 131:47–61, 1996.

[28] Julia Khodor. Design and implementation of computational systems based on programmed mutagenesis. Master's thesis, Massachussetts Institute of Technology, 1998.

[29] Julia Khodor and David K. Gifford. The efficiency of sequence-specific separation of DNA mixtures for biological computing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 39–46, 1999.

[30] K. Komiya, K. Sakamoto, H. Gouzo, S. Yokoyama, M. Arita, A. Nishikawa, and M. Hagiya. Successive state transitions with I/O interface by molecules. In *Lecture Notes in Computer Science, 2054, DNA Computing, A. Condon, G. Rozenberg (eds.)*, pages 17–26, 2001.

[31] H.M. Kong, R.B. Kucera, and W.E. Jack. Characterization of a DNA-polymerase from the hyperthermophile archaea thermococcus-litoralis - Vent DNA-polymerase, steady-state kinetics, thermal-stability, processivity, strand displacement, and exonuclease activities. *Journal of Biological Chemistry*, 268:1965–1975, 1993.

[32] T. H. LaBean, H. Yan, J. Kopatsch, F. Liu, E. Winfree, H.J. 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.

[33] Thomas H. LaBean, Erik Winfree, and John H. Reif. Experimental progress in computation by self-assembly of DNA tilings. In *DIMACS Series in Dis-*

crete Mathematics and Theoretical Computer Science, volume 54, DNA Based Computers V, E. Winfree and D.K. Gifford (eds.), pages 123–140, 2000.

[34] Michail G. Lagoudakis and Thomas H. LaBean. 2D DNA self-assembly for satisfiability. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 54, DNA Based Computers V, E. Winfree and D.K. Gifford (eds.)*, pages 141–154, 2000.

[35] L.F. Landweber, A.G. Fiks, and W. Gilbert. The boundaries of partially edited cytochrome c oxidase III transcripts are not conserved in kinetoplastids: implications for the guide RNA model of editing. *PNAS USA*, 90:9242–9246, 1993.

[36] Elizabeth Laun and Kalluru J. Reddy. Wet splicing systems. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 73–84, 1999.

[37] Xiaojun Li, Xiaoping Yang, Jing Qi, and Nadrian C. Seeman. Antiparallel DNA double crossover molecules as components for nanoconstruction. *Journal of the American Chemical Society*, 118(26):6131–6140, 1996.

[38] Y. Li and R.R. Breaker. Phosphorylating DNA with DNA. *PNAS USA*, 96:2746–2751, 1999.

[39] R.J. Lipton. DNA solutions of hard computational problems. *Science*, 268:542–545, 1995.

[40] Qinghua Liu, Anthony G. Frutos, Liman Wang, Andrew J. Thiel, Susan D. Gillmor, Todd Strother, Anne E. Condon, Robert M. Corn, Max G. Lagally, and Lloyd M. Smith. Progress toward demonstration of a surface based DNA computation: a one word approach to solve a model satisfiability problem. *BioSystems*, 52:25–33, 1998.

[41] Qinghua Liu, Andrew J. Thiel, Anthony G. Frutos, Robert M. Corn, and Lloyd M. Smith. Surface-based DNA computation: Hybridization and destruc-

tion. In *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23 - 25, 1997*, page 239, 1997.

[42] 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.

[43] Amit Marathe, Anne E. Condon, and Robert M. Corn. On combinatorial DNA word design. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 54, DNA Based Computers V, E. Winfree and D.K. Gifford (eds.)*, pages 75–90, 2000.

[44] M.L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., 1967.

[45] James C. Mitchell and Bernard Yurke. DNA scissors. In *Proceedings 7th DIMACS Workshop on DNA Based Computers, Tampa, FL, USA, June 2001*, pages 208–218, 2001.

[46] Nobuhiko Morimoto, Masanori Arita, and Akira Suyama. Solid phase DNA solution to the Hamiltonian Path problem. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 193–206, 1999.

[47] Gh. Paun, G. Rozenberg, and A. Salomaa. *DNA Computing: New Computing Paradigms.* Springer-Verlag, 1998.

[48] Gheorghe Paun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[49] Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. Restricted use of the splicing operation. Technical Report TR95-16, Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands, June 1995.

[50] J. Reif. Local parallel biomolecular computation. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 48, DNA Based Computers III, H. Rubin and D.H. Wood (eds.)*, pages 217–254, 1999.

[51] S. Roweis, E. Winfree, R. Burgoyne, N.V. Chelyapov, M.F. Goodman, P.W. Rothemund, and L.M. Adleman. A sticker-based model for dna computation. *J. Comput Biol*, 5(4):615–629, 1998.

[52] K. Sakomoto, H. Gouzo, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya. Molecular computation by DNA hairpin formation. *Science*, 288:1223–1226, 2000.

[53] K. Sakomoto, D. Kiga, K. Komiya, H. Gouzo, S. Yokoyama, S. Ikeda, H. Sugiyama, and M. Hagiya. State transitions by molecules. *BioSystems*, 52:81–91, 1998.

[54] J. SantaLucia, Jr., H.T. Allawi, and P.A. Seneviratne. Improved nearest-neighbor parameters for predicting DNA duplex stability. *Biochemistry*, 35(11):3555–3562, 1996.

[55] John SantaLucia, Jr. Personal communication.

[56] Lloyd M. Smith, Robert M. Corn, Anne E. Condon, Max G. Lagally, Anthony G. Frutos, Qinghua Liu, and Andrew J. Thiel. A surface-based approach to DNA computation. *Journal of Computational Biology*, 5(2):255–267, 1998.

[57] G.A. Soukup and R.R. Breaker. Engineering precision RNA molecular switches. *PNAS USA*, 96:3584–3589, 1999.

[58] A. Sreedhara, Y. Li, and R.R. Breaker. Ligating DNA with DNA. *Nature, submitted*, 2001.

[59] J. Tang and R.R. Breaker. Structural diversity of self-cleaving ribozymes. *PNAS USA*, 97(11):5784–5789, May 2000.

[60] A. J. Turberfield, Bernard Yurke, and Jr. A. P. Mills. DNA hybridization catalysts and molecular tweezers. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 54, DNA Based Computers V, E. Winfree and D.K. Gifford (eds.)*, pages 171–182, 2000.

[61] A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42:230–265, 1936.

[62] S. Tyagi and F.R. Kramer. Molecular beacons: Probes that flouresce upon hybridization. *Nature Biotechnology*, 14:303–308, 1996.

[63] Hiroki Uejima, Masami Hagiya, and Satoshi Kobayashi. Horn clause computation by self assembly of DNA molecules. In *Proceedings 7th DIMACS Workshop on DNA Based Computers, Tampa, FL, USA, June 2001*, pages 63–74, 2001.

[64] Kenichi Wakabayashi and Masayuki Yamamura. A realization of information gate by using enterococcus faecalis pheromone system. In *Proceedings 7th DIMACS Workshop on DNA Based Computers, Tampa, FL, USA, June 2001*, pages 199–207, 2001.

[65] R. Weiss and Jr. T. F. Knight. Engineered communications for microbial robotics. In *Lecture Notes in Computer Science, 2054, DNA Computing, A. Condon, G. Rozenberg (eds.)*, pages 1–16, 2001.

[66] H. Werntges, H.J. Fritz, D. Reisner, and G. Steger. Mismatches in DNA double strands– thermodynamic parameters and their correlation to repair efficiencies. *Nucleic Acids Research*, 14(9):3773–3790, 1986.

[67] J.G. Wetmur. DNA probes– applications of the principles of nucleic acid hybridization. *Critical Reviews in Biochemistry and Molecular Biology*, 26(3-4):227–259, 1991.

[68] E. Winfree. Algorithmic self-assembly of DNA: Theory and experiment. Talk at MIT, April 1998.

[69] E. Winfree, T. Eng, and G. Rozenberg. String tile models for DNA computing by self-assebly. In *Lecture Notes in Computer Science, 2054, DNA Computing, A. Condon, G. Rozenberg (eds.)*, pages 63–88, 2001.

[70] Erik Winfree. On the computational power of DNA annealing and ligation. In *DNA Based Computers*, pages 199–210, 1995.

[71] Erik Winfree. Complexity of restricted and unrestricted models of molecular computation. In *DNA Based Computers:DIMACS Workshop, April4, 1995*, pages 187–198, 1996.

[72] Erik Winfree, Furong Lin, Lisa A. Wenzler, and Nadrian C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–545, August 1998.

[73] Erik Winfree, Xiaoping Yang, and Nadrian C. Seeman. Universal computation via self-assembly of DNA: Some theory and experiments. In *Proceedings of the Second Annual Meeting on DNA Based Computers, held at Princeton University, June 10-12, 1996*, pages 172–190, 1996.

[74] M.A. Zenkova and G.G. Karpova. Imperfectly matched nucleic acid complexes and their biochemical manifestation. *Uspekhi Khimii (PNAS Russia)*, 62(4):414–435, 1993.