

Towards Programmable Molecular Machines

Ho-Lin Chen *

California Institute of Technology

Anindya De †

Indian Institute of Technology, Kanpur

Ashish Goel ‡

Stanford University

Abstract

Designing a programmable molecular machine is a fundamental problem in nano-technology. DNA is a good candidate for building these machines due to its small size and combinatorial nature. One experimentally promising direction towards obtaining such machines is a DNA walker, which is a DNA structure that moves along some pre-assembled substrate. A DNA walker can also be a useful tool for controllable molecular transportation. In this paper, we propose and analyze DNA walkers that can simulate arbitrary Turing machines within expected linear (in the execution length of the Turing Machine) time. We also develop a formal model and proof techniques for analyzing DNA walkers. Specifically, we define two classes of “safety properties”, and prove that a walker will operate correctly and efficiently, even in the presence of adversarial operations which disrupt the system, as long as it satisfies these properties, which our walkers do. Our walker consumes much less energy than previous proposed designs (linear as opposed to quadratic in the running time of the Turing machine being simulated), and is the first that can directly simulate arbitrary Turing Machines and the first that satisfies the safety properties.

*Ho-Lin Chen is at the Center for The Mathematics of Information, California Institute of Technology. This research was done when Ho-Lin was a student at Stanford University. Email: holin@stanford.edu. Research supported in part by NSF Award 0323766 and Caltech Center for Mathematics of Information.

†Anindya De is at the Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur, India. Most of this research was carried out while this author was at Stanford, and was supported by NSF grants 0524783 and 0650058. Email: anindya@cse.iitk.ac.in.

‡Ashish Goel is at the Department of Management Science and Engineering and (by courtesy) Computer Science, Stanford University, Terman 311, Stanford CA 94305. Email: ashishg@stanford.edu. Research supported by an NSF Career grant, NSF grants 0524783 and 0650058, and by the Stanford-KAUST alliance.

1 Introduction

A programmable and scalable nano-machine can potentially have a tremendous impact on both engineering tasks and medical applications. DNA is a good candidate for building programmable nano-machines due to its small size and “combinatorial” nature – you can change the sequence of a DNA molecule without affecting its double helical structure. Many nano-machines based on DNA have been proposed and demonstrated [16, 2, 6, 7, 10, 14, 13].

We believe theoretical techniques and algorithmic analysis can offer significant insight for the development of these machines. Our goal in this paper is to take one class of molecular machines and formally show how they can be used to perform programmable tasks. Before describing our results in more detail, we will summarize some recent related research.

Recent related results: Several experimental molecular machines have been recently developed in a laboratory setting. These machines are usually powered either by a natural phenomenon called strand invasion (also known as branch migration) or by enzymes that can break the backbone of DNA at locations specified by the DNA sequence itself (restriction enzymes) and link two DNA backbones together (ligation enzymes). Many of these machines are self-assembled and several of them move along a 1- or 2- dimensional self-assembled substrate. In each of these designs, each machine is a single (or a small number of) DNA molecule. Many copies of the machine are present in solution along with other enzymes or DNA strands needed for the machine to function.

One important DNA nano-machine is the DNA tweezer developed by Yurke *et al.* [16]. It has two rigid “arms” formed by double-stranded DNA helices connected by a short segment of single-stranded DNA. This machine can switch between two states “open” and “close” controlled using the strand invasion mechanism. This machine makes one state change every time a specific DNA strand is added. Later on, this construction was modified by Bishop *et al.* [2] into a DNA tweezer powered by enzymes; the operation of the machine consumes another type of DNA strand which they call fuel strand. This new tweezer switches between the two states autonomously until all the fuel strands get used up. Conceptually, this makes it quite similar to an engine, though of course such molecular machines can not yet be used to perform useful work.

There are several preliminary experiments for another DNA nano-machine called a DNA walker. A DNA walker is a DNA structure that moves along some pre-assembled substrate. It can be a useful tool for molecular transportation. Two proposed designs of such walkers are powered by strand invasion [6, 7]. In these proposals, the movement of the walker can be controlled, but two strand additions must be performed for each walker movement.

Yin *et al.* [15] proposed and implemented a DNA walker powered by enzymatic restrictions and ligations; this is the DNA walker that is most directly relevant to our results. In this implementation, there is a DNA substrate that acts as a “track” for the walker. There are several “anchorage” along the track. A double stranded DNA sequence (with one strand slightly longer than the other) is attached at each anchorage; thus at each anchorage, parts of the longer strand are exposed. The exposed part has a sequence which will allow the walker to bind to the next anchorage using complementary base pairing. The walker is represented by two DNA strands initially attached on top of the first anchorage. Again, one of the strands in the walker is partially exposed, and this part allows the walker to also attach to the next anchorage so that there are no exposed regions left at the original anchorage or the new anchorage. After the attachment, one enzyme ligates (i.e. attaches) the two strands together, so that there is no “break” along the DNA sequences. A second enzyme (a restriction enzyme) recognizes the newly formed continuous sequence, and makes a cut. After the cutting, everything remains the same except the walker strands have now moved to the top of the sec-

ond anchorage. This process can repeat by itself and the walker moves along the DNA track autonomously until it reaches the end. This was the first walker that could move autonomously without any environmental changes.

There are several important features of this design that deserve mention. The walker itself is only about 2nm long. This is orders of magnitude smaller than machines which can be fabricated with current technology, and even orders of magnitude smaller than simple organisms such as viruses. The movement of the walker is powered by enzymatic activity. Since these enzymes occur naturally in biological cells, they in turn can draw energy from well known bio-chemical processes (such as the ADP-ATP cycle) and do not require the development of new techniques. Most excitingly, the basic steps are very algorithmic in nature, raising the possibility that these steps may be strung together to perform complex operations in a programmable fashion. Such programmable molecular machines will be a new and powerful engineering primitive.

Tian *et al.* [10] also proposed a walker using DNAzymes. In this construction, the walker is a DNAzyme. It attaches to single stranded DNAs anchored on the substrate, makes a cut and moves onto the next DNA strand. Pei *et al.* [4] extended the DNAzyme walker to a 2-dimensional walker. In their design, the walker has multiple DNAzymes that can attach to DNA strands anchored on a 2-dimensional substrate. These DNAzymes can cleave the current DNA strand they bind to and non-deterministically move to the next one.

The walkers we have described so far are not programmable. These walkers just repeat the same reaction until there are no more places to move to. On the other hand, there were many interesting chemical and biological systems designed to simulate arbitrary Turing machines. Examples include self-assembly of DNA tiles [11], Building automata using enzymatic restrictions [1, 9] and general chemical and biological reaction networks [8, 3]. However, none of these systems can be easily adapted to program and control nano machines that does active work.

There have also been interesting proposed experiments for programmable walkers, though to the best of our knowledge, these experiments have not yet been conducted. Building upon the DNAzyme walker, Reif *et al.* [5] proposed a walker that simulates a restricted class of finite automata. Their construction encodes the finite automata on the substrate using different DNAzymes. The walker is a single stranded DNA encoding the input string plus some protecting structures. The walker will attach to the DNAzymes on the substrate. The DNAzyme can cleave the walker strand and this will reveal the next single-stranded portion of the walker and allow the walker to attach to the next DNAzyme. In their proposal, the walker moves on the substrate in a way corresponding to the computation made by the finite automata. The number of DNAzymes required in the construction is proportional to the number of symbols in the input string. Unfortunately, this design can only work when the transition table of the finite automaton can be laid out on a planar structure.

Yin *et al.* [14, 13] designed a walker that simulates a 2-state 5-symbol universal Turing machine and a walker that simulate a 2-state 2-symbol universal cellular automata. Both constructions use a constant number of restriction enzymes (4 and 3, respectively). They have provided detailed sequence designs and specified the enzymes they use. They also gave some preliminary evidence of the correctness of their scheme by showing that no unexpected reactions can block the normal operations forever. However, the amount of energy consumed by their walker is at least quadratic to the number of simulation steps due to reactions of different fuel strands. Also, the time required for the machine to operate is undiscussed. Their construction is tailored to a specific Turing machine. While that TM is universal, there is an extra layer of inefficiency in simulating a general TM using a universal Turing machine. Our results can be viewed as an extension of the same theme, where we develop a formal framework and proof techniques for proving correctness of such walkers, and present (new) walker designs that are provably correct and can simulate arbitrary Turing

Machines (TMs) or Deterministic Finite Automata (DFAs).

Our results: In this paper, we propose autonomous DNA walkers that can simulate arbitrary Turing machines within expected $O(n)$ time, where n is the number of operations for the Turing machine. This shows that our walker can do many interesting operations such as copying, counting, and pattern recognition. Our proposal extends the walker proposed by Yin *et al.* [15] and provides a method for controlling the movement of the walker and for the walker to take input from (and write output to) the substrate. Some of the techniques we use are similar to the ones described in [13]. In our proposal, the substrate is a long 1-dimensional track with many anchorages. Each symbol of the input string is encoded on the corresponding anchorage on the substrate and the walker contains the internal state of the machine. The walker can move forwards and backwards on the substrate autonomously and perform the computation done by the Turing machine, leaving the correct output on the substrate at the end. Our construction only uses 2 restriction enzymes to simulate arbitrary finite automata and 3 restriction enzymes to simulate arbitrary Turing machines¹. The transition tables of the automata/Turing machines are encoded by DNA strands introduced in solution, whereas the enzymes/substrate design do not depend on details of the specific automaton/Turing machine being simulated.

Equally importantly, we develop a model and proof techniques to formally show the correctness of our walkers. Under this model, we can show that our walker always completes correctly and the expected computation time and energy consumption are both linear to the Turing machine steps. We first define the basic operations of our model – complementary pairing, cleavage activity by restriction enzymes, joining of two DNA strands using ligation enzymes, and the detachment of two DNA strands held together by insufficient strength. We then define adversarial operations that are significantly more powerful than the normal operations – for instance, strands which are permanently held together if only normal operations are allowed may get detached under adversarial operations. The adversarial model is intended to model errors or other unexpected phenomena; such errors have been known to plague DNA self-assembly systems [12]. We then define two classes of “safety properties”, and prove that these safety properties imply that the walker will operate in linear time (if strong safety properties are satisfied) or eventually (if only weak safety properties are satisfied), *even in the presence of adversarial operations which disrupt the system*. Finally, we show that the walker designs we propose satisfy the strong safety properties, and hence, can operate in expected time which is linear in the number of operations. In a walker system, each reaction takes a certain amount of energy to complete and therefore we would like to minimize the number of reactions required. For all of our walker constructions, we show that the expected number of reactions happening in the system is proportional to the number of walker steps w whereas all previous constructions require at least $O(w^2)$ number of reactions². We believe that our proof technique will be useful to other nano-machine constructions.

We will define our walker model and the safety properties in section 2. Then, we will describe our walker design that simulates DFAs and prove its correctness (section 3). We will do the same for TMs in section 4. While the DFA walker is of course subsumed by the TM walker, we believe that seeing the DFA walker first will help the reader build more intuition about the TM walker. Finally, we describe several important open problems in section 6.

¹The restriction enzymes we use in our paper are not actual known physical enzymes. Rather, they are reasonable abstractions of known real restriction enzymes.

²If the walker can move w steps, then to ensure proper walker operation, the amount of fuel strand must be at least w times the amount of machines. Therefore, if the fuel strands can react with each other, as in [14, 13], it will happen w times more frequently than the operations which helps the walker move and the total number of reactions between fuel strands must be at least $O(w^2)$.

2 Model

Our theoretical model extends the walker model proposed and implemented by Yin *et al.* [15].

In our model, there is a linear substrate acting as a track. There are some DNA molecules anchored to this substrate with fixed spacings between them and some other DNA molecules floating in the solution. We use the term *anchorage* to denote DNA molecules anchored to this substrate. All DNA molecules are duplex DNA molecules with some single stranded regions that can be used to attach to molecules and hence we can specify a molecule just by specifying its sequence. For the ease of description, we use abstract symbols to denote the DNA sequences for the rest of the paper. All symbols represent a DNA sequence of length $L = O(\log |S|)$, where $|S|$ is the total number of strands. Functions ϕ and θ map states or symbols to DNA sequences of length L . Every symbol represents a different DNA sequence and differ by at least $L/2$ bases unless otherwise specified. Adding the superscript r to a symbol denotes the reverse of the sequence and adding the superscript c denotes the complement of the sequence. Characters with underlines and overlines denote single stranded portions of a molecule; other characters denote double stranded regions. The characters are listed from the 3' end to the 5' end. An example is shown in figure 1(a). It is worth noticing that each molecule has two different ways of describing. For example, a molecule denoted by $A \underline{B} C$ is exactly the same as a molecule denoted by $C^{rc} \overline{B^{rc}} A^{rc}$. In the rest of this paper, we will describe the sequences of the anchorages starting from the anchored end and describe other DNA molecules according to the direction of the molecule when they are about to attach to an anchorage.

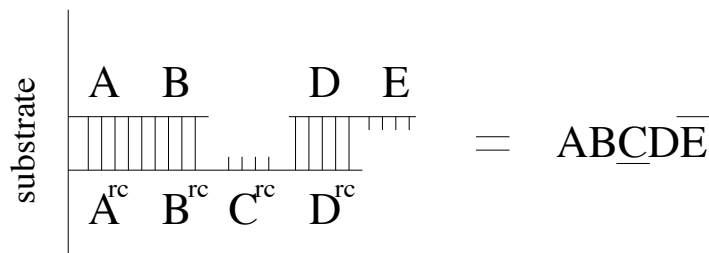


Figure 1: (a) The anchored strand illustrated in this figure is denoted by $ABC\underline{D}\overline{E}$. Superscripts c and r denote the complementary strand and the reversed strand respectively. (b)

Initially, the walker strand attaches on the first anchored strand. The walker is going to move autonomously along this substrate and perform some computation. During the motion, some part of the walker strand can get replaced by the strands floating in the solution in order to encode information for computational purposes. We only consider four types of reactions. The first one is *hybridization*, which is attachment of two molecules having complementary single stranded regions of length at least c . We also consider the reverse of this reaction which we call *detachment*. This corresponds to the falling apart of two DNA strands which are held by at most c base pairs. The other two type of reaction are ligation and enzymatic restriction. In an ligation reaction, DNA ligase links together DNA strands that has a break in both complementary strands of DNA; in a enzymatic restriction, an enzyme recognizes a special double stranded sequence (called *restriction site*) and makes two incisions, one through each of the sugar-phosphate backbones at some location with a fixed distance to the restriction site. In our notation, when two molecules of the form $A\overline{B}$ and $\underline{B}C$ attach (hybridize) together, ligation can link them into a single molecule ABC and restriction can cut a molecule of the form ABC into $A\overline{B}$ and $\underline{B}C$ but can only make the cut at a fixed location with respect to the restriction site. We also assume that the restriction site can be two regions separated by a fixed distance, since this type of enzyme exists in nature and is used in previous constructions [14, 13].

We define *normal* operations to be the set of all possible reactions of the above four type. We also define a larger set of reactions called the *adversary* operations. The adversary operations are the same as the normal operations except two DNA strands can hybridize when they have at least $c/2$ complementary base pairs and two DNA strands can detach if they are held by less than $2c$ base pairs. For the normal walker operations, we adjust the concentration of each molecule and the environment such that all four types of normal operations happen at the same expected rate. Note that the adversary needs only half as much strength to attach two molecules and can detach molecules held by twice as much strength comparing to normal operations. Since the rate of the reaction depends exponentially on the energy change, we know that all reactions that are not adversary operations happen with a much slower rate and in the rest of this paper, we assume that such reactions do not take place. We also set the constant threshold c to be the length L , which is the length of a sequence corresponding to every abstract symbol.

To show that a walker will perform a certain set of operations as designed, we need to define the following terms to describe the state of a walker.

Definition 1 *The configuration of a walker is defined by the sequence of all anchorages.*

Definition 2 *The c_1 -neighborhood of a configuration P is the set of all walker configurations from which configuration P can be reached with at most c_1 normal operations on each anchorage.*

For our purposes, c_1 will be a constant that will depend on the construction. We also ignore all configurations that are not reachable from the initial configuration. To ensure that the walker can function in a way we expect, we must define a series of desired walker configurations P_1, P_2, \dots, P_k . We describe two sets of properties which we call *safety* and *strong safety*. We show that when all safety properties are satisfied, the system will reach configuration P_k eventually and when all strong safety properties are satisfied, the system will reach some configuration similar to P_k in $O(kT)$ time, where T is the expected time for a normal operation to take place.

Definition 3 *The following properties are called Safety properties.*

1. *configuration P_i can be reached from configuration P_{i-1} with c_2 operations where c_2 is a constant.*
2. *Starting from configuration P_i , no matter how many adversary operations we do, the configuration of the walker will be in the c_1 -neighborhood of state P_j for some $j \geq i$, where c_1 is a constant.*

Intuitively, the first property guarantees the existence of a series of reaction that takes the walker from the initial state to the desired final state. The second property guarantees that the current configuration of the walker cannot be very different from a desired configuration and there is always a way to undo all the unwanted reactions.

Definition 4 *The following properties are called Strong safety properties.*

1. *All safety properties.*
2. *In all configurations reachable with adversary operations, when two molecules attach together, they are held by at least c base pairs.*
3. *At any given configuration of the walker, the number of adversary operation that can be performed on a specific anchorage is bounded by a constant, c_3 .*

4. From any configuration in the c_1 -neighborhood of P_{i-1} , the walker can reach a configuration in the c_1 -neighborhood of P_i within c_4 normal operations, where c_4 is a constant.

In the rest of the section, We describe our main theoretical results on the correctness of walker designs. First, we show that when all safety properties are satisfied, the configuration P_k can be reached eventually.

Theorem 1 *After an adversary makes any number of adversary operations starting from configuration P_1 , configuration P_k can be reached using only normal operations.*

Proof: From the second property of safety, we know that any configuration reachable from P_1 must be in the c_1 -neighborhood of P_i for some i . Therefore, configuration P_i can be reached using only normal operations. Using the first property of safety, configuration P_k can be reached from P_i using only normal operations. \square

Next, we show that when all strong safety properties are satisfied, we can reach a configuration in the neighborhood of P_k in $O(k)$ time. We start by showing that we only need a small number of normal operations to reach the neighborhood of P_k .

Theorem 2 *When all strong safety properties are satisfied and the walker starts from any configuration in the c_1 -neighborhood of P_i for some i , the walker will reach a configuration in the neighborhood of configuration P_k within expected $O(k)$ time.*

Proof: From the fourth property of strong safety, we know that from any configuration in the c_1 -neighborhood of P_j , we need at most c_4 normal reactions to reach a configuration in the c_1 -neighborhood of P_{j+1} . There are at most $2c_4$ anchorages involved in these c_4 reactions. From the third property of strong safety, we know that at most $2c_3c_4$ possible adversary operations can happen to these anchorages at any given time. From the second property of strong safety, we know that these c_4 normal reactions happen with the highest rate among all adversary operations. Therefore, when one of these c_4 reactions is available, it will happen before any other adversary operation on these anchorages with probability at least $\frac{1}{2c_3c_4}$. Hence, there is a constant probability that these c_4 reactions happen before any other reaction on these $2c_4$ anchorages. Notice that there could be other adversary operations happening on other anchorages during this time.

If all c_4 reactions happen before any other reactions on these anchorages, we can consider the configuration P^* of the walker right after all these reactions happen. Notice that there could be other adversary operations happening on other anchorages during this time and therefore P^* is not necessarily in the c_1 -neighborhood of P_{j+1} . However, since any other reactions doesn't affect the c_4 operations we consider, we know that P_* can be reached by performing those c_4 operations first followed by all other adversary operations. This shows that P_* is reachable from some configuration in the c_1 -neighborhood of P_{j+1} and therefore must be in the c_1 -neighborhood of P_k for some $k > j$ by the definition of "neighborhood" and the second property of safety. From the above, we know that starting from any state in the c_1 -neighborhood of P_j , we will be in the c_1 -neighborhood of P_k for some $k > j$ with expected constant number of reactions and this proves the theorem. \square

3 Simulating Deterministic Finite Automata

In this section, we will present a walker system that simulates a finite automata. The input string is encoded on the anchorages on the substrate. The i -th symbol is encoded on the i -th anchorage by adding a restriction site at a specific location. The walker encodes the current state of the automata by encoding a transition table from the current state to the next. In each cycle of the operation, a "read" is performed by an enzyme which

recognizes the restriction site and reveals a subsequence of the walker that encodes the next state. Several reactions happen according to the sequence revealed. After these reactions, a walker strand encoding the next state will be attached to the next anchorage which completes a cycle. This walker uses only three enzymes, including one enzyme that does ligation.

3.1 System Description

For any given finite state automaton with state space $S = \{S_0, S_1, S_2, \dots, S_k\}$ and symbol space $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_t\}$. Let the input string be $a_1 a_2 \dots a_n$, and initial state be S_0 . Also, we use $NS_i(j)$ to denote the next state of the walker if the current state is S_j and input symbol is σ_i . For the ease of description, we use \square_j to denote $\underbrace{JJ \dots J}_{j-1} R_1 \underbrace{JJ \dots J}_{t-j}$ for the rest of this section.

The substrate has $n + 1$ anchorages on it with equal spacing between every two adjacent strands. For every location $u > 1$, let $a_u = \sigma_j$. Then anchorage at location u of the substrate is $\square_j \underline{A}$. The $n + 1$ -th anchorage is $\underbrace{JJ \dots J}_t \underline{A}$. Let the first symbol be σ_i . The first strand (corresponding to the anchorage at the initial position of the head and the walker with state S_0) is

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \dots \phi_t(NS_t(0)) \underbrace{JJ \dots J}_{t+3} \underbrace{CC \dots C}_{t+2}.$$

We also add the following four types of DNA molecules in the solution:

$$\underline{C} \underbrace{JJ \dots J}_{t+1} R_1^{rc}.$$

$$\overline{\phi_i(S_j)} \underbrace{JJ \dots J}_{t-i} R_2^{rc} \theta(S_j) A^{rc} K \underbrace{JJ \dots J}_{t-1} D_1 D_2 \dots D_t J J, \text{ for every symbol } \sigma_i \text{ and state } S_j.$$

$$\underline{D_j} D_{j+1} \dots D_t \underbrace{JJ \dots J}_{t+4} R_1^{rc}, \text{ for every } 1 \leq j \leq t.$$

$$\overline{\theta(S_j)^{rc}} \phi_1(NS_1(j)) \phi_2(NS_2(j)) \dots \phi_t(NS_t(j)) \underbrace{JJ \dots J}_{t+3} \underbrace{CC \dots C}_{t+2}, \text{ for every state } S_j.$$

There are three enzymes in this system. Each of these enzymes has a specific restriction site and makes one cleave on each of the two DNA strands. The first enzyme E_1 recognizes a double stranded region R_1 . The cleavage on the strand with sequence R_1 is $(3t + 5)L$ bases from the restriction site R_1 and the cleavage on the complementary strand is $(3t + 4)L$ bases away from the restriction site R_1^{rc} . The second enzyme E_2 recognizes $R_2^{rc} X A^{rc} J^{rc}$, where X is any sequence of length L . This enzyme cuts the double stranded region into $R_2^{rc} \overline{X}$ and $\underline{X} A^{rc} J^{rc}$. We also have an enzyme which does ligation.

3.2 Desired Sequence of Reactions

We now describe how this walker moves and simulates any given automata.

Step 1:

In the initial configuration, enzyme E_1 can recognize R_1 on the first anchorage

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \dots \phi_t(NS_t(0)) \underbrace{JJ \dots J}_{t+3} \underbrace{CC \dots C}_{t+2}$$

and make a cleavage. After this, only the bottom part of this anchorage

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_t(NS_t(0)) \underbrace{JJ \cdots J}_{t+3} \underbrace{CC \cdots C}_{i-1} \overline{C}$$

is anchored. The other part can detach and drift away.

Step 2:

The strand $\underline{C} \underbrace{JJ \cdots J}_{t+1} R_1^{rc}$ can now attach using the complementary sequence \overline{C} and \underline{C} and be ligated to the anchored strand. After the ligation, we have

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_t(NS_t(0)) \underbrace{JJ \cdots J}_{t+3} \underbrace{CC \cdots C}_{i-1} C \underbrace{JJ \cdots J}_{t+1} R_1^{rc}$$

The enzyme E_1 can now recognize either R_1 or R_1^{rc} in this anchored DNA sequence. If it recognizes R_1 and makes the cleavage first, the configuration goes back to the end of Step 1. If it recognizes R_1^{rc} , the anchorage will become

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \underline{\phi_i(NS_i(0))}$$

Step 3:

Notice that $\overline{NS_i(0)}$ is the next state of the walker. We assume that $NS_i(0) = S_j$ for $j \in \{1, 2, \dots, k\}$.

The strand $\overline{\phi_i(S_j)} \underbrace{JJ \cdots J}_{t-i} R_2^{rc} \theta(S_j) A^{rc} K \underbrace{JJ \cdots J}_{t-1} D_1 D_2 \cdots D_t J J$ can attach to the anchored strand produced by step 2. After this strand attaches and gets ligated, enzyme E_1 can function again and produce the following anchorage.

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_i(NS_i(0)) \underbrace{JJ \cdots J}_{t-i} R_2^{rc} \theta(S_j) A^{rc} K \underbrace{JJ \cdots J}_{t-1} D_1 D_2 \cdots D_{i-1} \overline{D_i}$$

Step 4:

The strand $\underline{D_j} D_{j+1} \cdots D_t \underbrace{JJ \cdots J}_{t+4} R_1^{rc}$ can attach to the anchorage produced by step 3 and, similar to step 2, the enzyme E_1 can recognize either R_1 or R_1^{rc} . If R_1 is recognized, the configuration goes back to the end of step 3. If R_1^{rc} is recognized, then the current anchorage become

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_i(NS_i(0)) \underbrace{JJ \cdots J}_{t-i} R_2^{rc} \theta(S_j) \underline{A^{rc}}$$

Step 5:

Now, the current anchorage can attach to the next anchorage on the substrate using complementary sequences \underline{A} and $\underline{A^{rc}}$. We get (assuming the next input symbol is σ_u)

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_i(NS_i(0)) \underbrace{JJ \cdots J}_{t-i} R_2^{rc} \theta(S_j) A^{rc} \square_u^{rc}$$

Both ends of this double stranded sequence are anchored on the substrate. After the attachment, enzyme E_2 will function and split this DNA sequence into the following two anchorages.

$$\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_i(NS_i(0)) \underbrace{JJ \cdots J}_{t-i} R_2^{rc} \overline{\theta(S_j)}$$

and

$$\square_u A \overline{\theta(S_j)^{rc}}$$

The strand $\overline{\theta(S_j)^{rc}} \phi_1(NS_1(j)) \phi_2(NS_2(j)) \cdots \phi_t(NS_t(j)) \underbrace{JJ \cdots J}_{t+3} \underbrace{CC \cdots C}_{t+2}$ can now attach to the second anchorage. This completes one cycle of the walker operation which corresponds to one step of the automata we want to simulate.

The following strands will be generated during the walker operations:

Step 1:

$$\overline{C} \underbrace{CC \cdots C}_i, 2 \leq i \leq t+1$$

Step 2:

$$\overline{\phi_i(NS_i(j))} \phi_{i+1}(NS_{i+1}(j)) \cdots \phi_t(NS_t(j)) \underbrace{CC \cdots C}_{i-1} C \underbrace{JJ \cdots J}_{t+1} R_1^{rc}, 1 \leq i \leq t, S_j \text{ is a state of the automata}$$

Step 3:

$$\overline{D_i} D_{i+1} \cdots D_t J J, 1 \leq i \leq t$$

Step 4:

$$\overline{A}^{rc} K \underbrace{JJ \cdots J}_{t-1} D_1 D_2 \cdots D_t \underbrace{JJ \cdots J}_{t+4} R_1^{rc}$$

This walker satisfies all safety property mentioned in section 2. The detailed proof is in the appendix due to space constraint.

As mentioned previously, this walker has another important feature - low energy consumption. The expected amount of reactions is proportional to the number of operations of the Turing machine simulated. To show this, first notice that no two molecules in the solution share common sticky ends. Therefore, no reaction is going to happen between molecules in the solution. Second, when the walker is on anchorage x , all anchorages $y, y < x$ have the form $\square_i A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_i(NS_i(0)) \underbrace{JJ \cdots J}_{t-i} R_2^{rc} \overline{\theta(S_j)}$

and all anchorages $y, y > x$ have the form $\square_j \underline{A}$. These anchorages don't react with any molecule in the solution. Therefore, all reactions must involve the anchorage with the walker and thus there are only a constant number of operations that can happen in the whole system at any given time. From the safety proofs, we know that the system finishes in expected $O(n)$ time where n is the number of operations of the Turing machine. Therefore, the expected total number of reactions is $O(nN)$ where N is the number of walkers we have in the experiment.

4 Turing Machine

In this section, we will present a walker that simulates a Turing machine. The input string is encoded on the anchorages on the substrate. The i -th symbol is encoded on the i -th anchorage by adding a restriction site at a specific location. The walker encodes the current state of the Turing machine by encoding a transition table from the current state to the next. In each cycle of the operation, a “read” is performed by an enzyme which recognizes the restriction site and reveals a subsequence of the walker that encodes the next state. Several reactions happen according to the sequence revealed. These reactions will move the walker to an adjacent anchorage (corresponding to a head move) and change the sequence of the leftover portion of the anchorage (corresponding to a “write” operation). The walker described in this section only uses 3 restriction enzymes. Figure 2 illustrates the idea of how the walker operates. The detailed construction of the walker and the proof of safety properties are in the appendix due to space constraints. In this construction, the expected number of operations is also linear to the number of operations for the Turing machine.

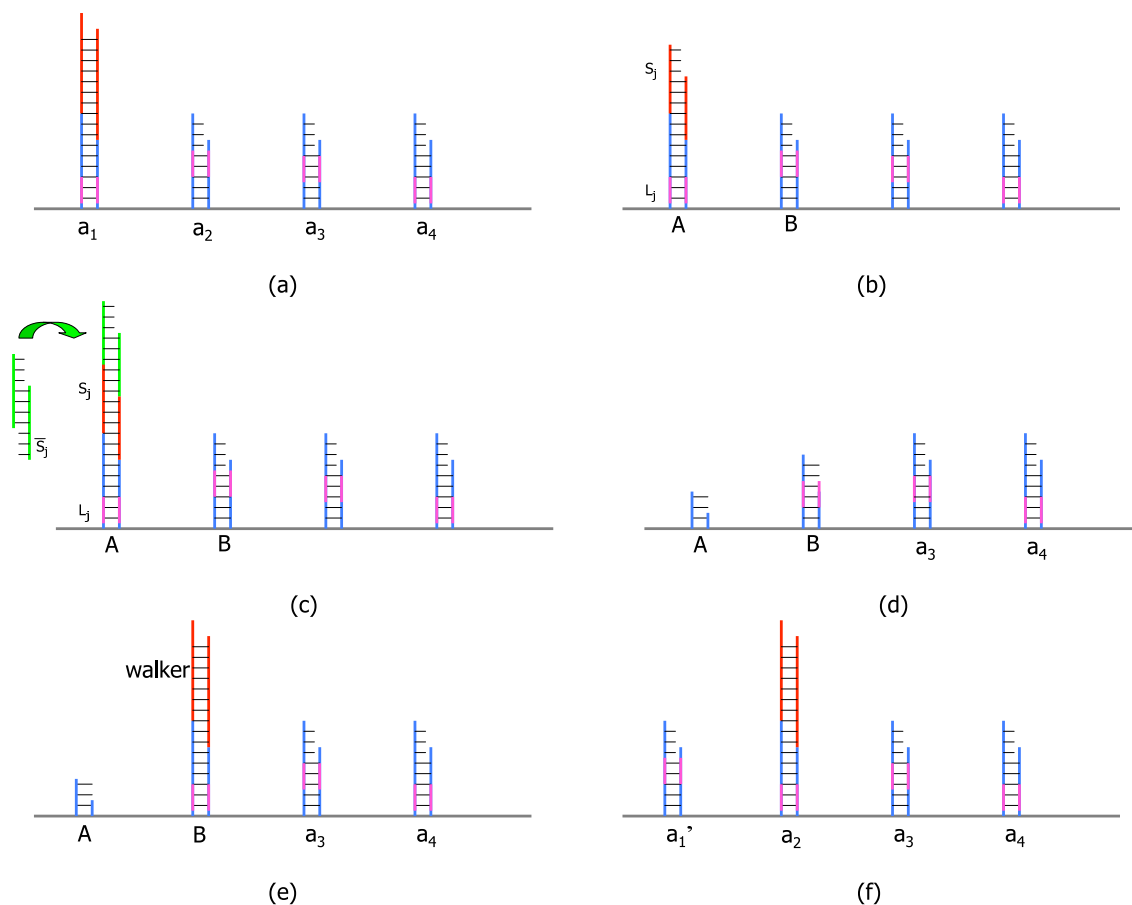


Figure 2: (a) The starting state. Red strands denote the DNA walker. (b) An enzyme cleaves the walker and reveals a sequence corresponding to the next state S_j . This corresponds to a “read” operation. (c) A sequence of reactions happen which allow the walker to bind to an adjacent anchorage B . (d) After the two anchorages A and B binds together, two enzymes can function and make cuts on both of the two anchorages. (e) One molecule encoding the next state can attach to anchorage B . (f) A series of reactions happen to anchorage A and changes the location of one special restriction site. This corresponds to a “write” operation.

5 Conclusion

In this paper, we show that we can extend the walker by Yin *et al.* [15] to simulate arbitrary finite automata and Turing machines. Our construction only uses 2 restriction enzymes for simulating finite automata and 3 restriction enzymes for simulating Turing machines. The specification of the Turing machine goes into the strands in the solution and anchored on the substrate. We also show that the walker can operate correctly unless some low-probability catastrophic event happens. However, our construction requires an enzyme that can count $O(|\Sigma|(\log |S| + \log |\Sigma|))$ bases, where Σ is the symbol space and S is the state space of the Turing machine we want to simulate. This could be very hard to implement if we want to simulate complicated Turing machines. It is an interesting open problem to construct controllable walkers using enzymes which count for shorter distances.

6 Open problems

There are several open problems which directly relate to our paper: *Can we reduce the number of enzymes? Are there designs that are robust against stronger adversaries? Can we reduce the length of sequences over which the restriction enzymes in our designs must count? And most importantly (though perhaps not of direct interest to the theory community), can these designs be verified experimentally?* We believe that these are all very important open problems deserving further study. However, we would also like to pose two other problems which are less directly related to the results of our paper but are of significant interest for molecular machines in general:

1. Earlier in this paper, we mentioned results [16, 6, 7] where strand invasion was used to perform operations that could also be done using restriction enzymes. This leads to the following natural question: *Can strand invasion be used to simulate general restriction and ligation enzymes?*
2. We need several restriction enzymes in our walkers, and these restriction enzymes need to count over long sequences. It is not immediately obvious that such enzymes occur in nature and can co-exist under the same experimental conditions. *Is there some way of simulating general restriction enzymes using a small number of simple restriction enzymes, perhaps in conjunction with strand invasion?*

All the above problems have a very algorithmic/computational flavor. We expect that any theoretical techniques developed for their solution will have direct impact on experimental capability.

References

- [1] Y. Benenson, R. Adar, T. Paz-Elizur, Z. Livneh, and E. Shapiro. DNA molecule provides a computing machine with both data and fuel. *Proc. Natl. Acad. Sci. USA*, (100):2191–2196, 2003.
- [2] J. Bishop and E. Klavins. Improved repeatability of an autonomous DNA nanomotor, 2007.
- [3] L. Cardelli and G. Zavattaro. On the computational power of biochemistry. In *Proceedings of the third International Conference on Algebraic Biology*. Austria, July 2008.
- [4] R. Pei, S. Taylor, and M. Stojanovic. Coupling computing, movement, and drug release, 2007.

- [5] J. H. Reif and S. Sahu. Autonomous programmable DNA nanorobotic devices using DNAzymes. In *Proceedings of the Thirteenth International Meeting on DNA Based Computers*. Memphis, TN, June 2007.
- [6] W.B. Sherman and N.C. Seeman. A precisely controlled DNA bipedal walking device. *NanoLetters*, 4:1203–1207, 2004.
- [7] J.-S. Shin and N.A. Pierce. A synthetic DNA walker for molecular transport. *Journal of American Chemistry Society*, 126:10834–10835, 2004.
- [8] D. Soloveichik, M. Cook, E. Winfree, and S. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, February 2008.
- [9] D. Soloveichik and E. Winfree. The computational power of benenson automata. *Theoretical Computer Science*, (244):279–297, 2005.
- [10] Y. Tian, Y. He, Y. Chen, P. Yin, and C. Mao. A DNAzyme that walks processively and autonomously along a one-dimensional track. *Angewandte Chemie*, 44:4355–4358, 2005.
- [11] E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, Pasadena, 1998.
- [12] E. Winfree and R. Bekbolatov. Proofreading tile sets: Error correction for algorithmic self-assembly. In *Proceedings of the Ninth International Meeting on DNA Based Computers*. Madison, Wisconsin, June 2003.
- [13] P. Yin, S. Sahu, A. J. Turberfield, and J. H. Reif. Design of autonomous DNA cellular automata. In *Proceedings of the Eleventh International Meeting on DNA Computing*. London, Ontario, Canada, June 2005.
- [14] P. Yin, A. J. Turberfield, S. Sahu, and J. H. Reif. Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion. In *Proceedings of the Tenth International Meeting on DNA Based Computers*. Milan, Italy, June 2004.
- [15] P. Yin, A. J. Turberfield, S. Sahu, and J. H. Reif. Designs for autonomous unidirectional walking DNA devices. In *Proceedings of the Tenth International Meeting on DNA Based Computers*. Milan, Italy, June 2004.
- [16] B. Yurke, A. Turberfield, A. Mills Jr, F. Simmel, and J. Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, (406):605–608, Aug 2000.

A Safety Properties for Finite Automata Simulation

Safety:

Ideally, we would like the walker to repeat the cycle mentioned in section 3.2 n times and stop. The walker with the final state encoded on it will be attached on top of the $n + 1$ -th anchorage. In the rest of this section, we show that the walker will perform all these desired operations by showing that all safety properties hold. We define configuration P_i to be the configuration at the beginning of the i -th cycle for

$1 \leq i \leq n$ and define P_{n+1} to be the desired final configuration. When the walker is at configuration P_i , then configuration P_{i+1} can be reached using one cycle. Therefore, the first property of safety is true.

To prove the system satisfies the second property of safety, first, it is easy to see that for all double-stranded molecules in this system, the two strands are held by more than $2L$ base pairs. Hence, the adversary cannot break any of them to generate new sequences. Also, by our initial specification of the sequences, we know that any two non-complementary strand can only have less than $L/2$ complementary base pairs. Hence, the set of adversary operations and normal operations are exactly the same. The following lemma shows that the walker satisfies the second property of safety.

Lemma 3 *Starting from configuration P_i , the configuration of the walker will remain in the 10-neighborhood of P_i until reaching configuration P_{i+1} .*

Proof: We know that the configuration P_i and P_{i+1} only differs on the i -th and the $i + 1$ -th anchorages. In these two configurations, the first $i - 1$ anchorages are of the form (without loss of generality, we call the current state S_0)

$$\square_u A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_u(NS_u(0)) \underbrace{JJ \cdots J}_{t-u} R_2^{rc} \overline{\theta(S_v)}$$

for some u, v and anchorages $i + 2$ to n are of the form

$$\square_u A$$

It is easy to see that no molecules floating in the solution can attach to any of these anchorages and therefore all these anchorages will remain unchanged unless they can bind to the i -th or $i + 1$ -th anchorage.

Hence, to prove the lemma, we only need to show in the reaction cycle described in section 3, all other possible reactions are reversible and cannot change the configuration too much.

Starting from configuration P_i , the only reaction that can happen is the enzymatic restriction described. Therefore, anchorage i can only change to

$$\square_u A \theta(S_0)^{rc} \phi_1(NS_1(0)) \phi_2(NS_2(0)) \cdots \phi_t(NS_t(0)) \underbrace{JJ \cdots J}_{t+3} \underbrace{CC \cdots C}_{u-1} \overline{C}$$

From this configuration, the only two sets of molecules that can attach are

$$\underline{C} \underbrace{JJ \cdots J}_{t+1} R_1^{rc}$$

and

$$\overline{C} \underbrace{CC \cdots C}_i, 2 \leq i \leq t + 1$$

As described in the step 2 of the walker cycle, if the first strand attaches, the configuration of the anchorage with walker can only change to the configuration described at the end of step 2 or remain at the configuration mentioned at then end of step 1. On the other hand, if the second strand attaches, no further reactions can happen until it gets cleaved by enzyme E_1 again and goes back to the situation at the end of step 1.

The situation for all other steps is the same. The adversary can attach and ligate one unexpected molecule to the anchorage with the walker at any time, but no further reaction can happen except the attached molecule

falling off (due to an enzymatic cleavage). Therefore, before the walker reaches configuration P_{i+1} , at most 10 normal operations are required to change the anchorage i to back to the sequence described by P_i and all other anchorages are identical to the one described by P_i . Therefore, we only need to show that from any of these configurations, P_{i-1} cannot be reached using any number of adversary operations.

From the above discussion, we know that the first $i - 1$ anchorages remain unchanged during the whole cycle no matter which adversary operations has happened. Using this argument on all other cycles, we know that the sequence of the i -th anchorage will never be changed again after reaching configuration P_{i+1} . Also, notice that for all configurations described in the cycle, the i -th anchorage always starts with sequence $\square_u A \theta(S_0)^{rc}$. Therefore, starting from the configuration P_i , anchorage i will always contain the sequence $\square_u A \theta(S_0)^{rc}$ no matter how many adversary operations happen and the configuration will never reach P_j for any $j < i$. \square

Strong safety:

The second property of strong safety follows from the fact that all adversary operations are normal operations. From lemma 3, we know that for every configuration of the walker, there are at most two adversary operations that can happen. We also know that for any configuration in the 10-neighborhood of P_i , all anchorages other than i and $i + 1$ have sequences identical to the sequence described in P_{i+1} and from the description of the walker cycle, we know that configuration P_{i+1} , which is in the neighborhood of P_{i+1} , can be reached in 10 operations simply by following the walker cycle described above. This shows the last property for strong safety is satisfied.

From the above discussion, we know that the walker we constructed can simulate any given finite automata and it only takes expected $O(n)$ reactions to simulate an automata with input length n .

B Details on Simulating Turing Machines

In this section, we will present a walker that simulates a Turing machine. The input string is encoded on the anchorages on the substrate. The i -th symbol is encoded on the i -th anchorage by adding a restriction site at a specific location. The walker encodes the current state of the Turing machine by encoding a transition table from the current state to the next. In each cycle of the operation, a “read” is performed by an enzyme which recognizes the restriction site and reveals a subsequence of the walker that encodes the next state. Several reactions happen according to the sequence revealed. These reactions will move the walker to an adjacent anchorage (corresponding to a head move) and change the sequence of the leftover portion of the anchorage (corresponding to a “write” operation).

B.1 System Description

For any given Turing machine, we define a new state space such that each state S_i consists of a four tuple (Turing machine state, symbol that should be written on current head position, direction of head movement, residue of current position to base 3). Let the new state space $S = \{S_1, S_2, \dots, S_k\}$ and the new symbol space $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_t\}$. Let the initial symbols on the tape be $a_1 a_2 \dots a_n \dots$, initial head position be 1 and current state be S_{ini} . Also, we use $NS_i(j)$ to denote the next state of the walker if the current state is S_j and input symbol is σ_i . We also assume $k > j$ for the rest of the section.

The substrate has n anchorages on it with equal spacing between every two adjacent strands. For every location $u > 1$, let $a_u = \sigma_j$ and w be the residue of u to base 3. Then anchorage at location u of the substrate is $\square_{w,j} \overline{A_w}$, where

$$\square_{w,j} = \phi_{1w}(\sigma_1) \phi_{1w}(\sigma_2) \cdots \phi_{1w}(\sigma_t) \underbrace{JJ \cdots J}_{j-1} R_1 \underbrace{JJ \cdots J}_{t-j} \phi_2(S_k) \phi_2(S_{k-1}) \cdots \phi_2(S_1) B$$

All sequences of the form $\phi_{1x}(\sigma_y)$, $x \in \{1, 2, 3\}$ and $1 \leq y \leq t$ end with a short sequence K , but any two of them still differs by at least $L/2$ bases, as previously stated. The first strand (corresponding to the anchorage at the initial position of the head and the walker with initial state S_1) is

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_t(NS_t(1)) \underbrace{JJ \cdots J}_{10k} \underbrace{CC \cdots C}_{t+2}$$

(assuming $a_1 = \sigma_i$)

The following strands are floating in the solution:

$$\overline{C} \underbrace{JJ \cdots J}_{t+k+1} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$$

$$\underline{\theta_i(S_j)} \underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{j-1} \phi_{4i}(S_j) \underbrace{JJ \cdots J}_{10k+i-j-2} C_{ij}, 1 \leq i \leq t, 1 \leq j \leq k$$

$$\overline{C_{ij}} \underbrace{JJ \cdots J}_{2t+k-i+j+2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}, 1 \leq i \leq t, 1 \leq j \leq k$$

$$\underline{\phi_{4i}(S_j)} R_2 \underbrace{JJ \cdots J}_{k-j-1} R_3 A_m^{rc} J D_1 D_2 \cdots D_j, 1 \leq i \leq t, 1 \leq j \leq k$$

(The value of m depends on the state S_j and will determine the direction which the walker moves. For example, if S_j indicates that the walker is currently on the $3u + 1$ -th anchored strand and should move towards right, then $m = 2$.)

$$\overline{D_j} D_{j+1} \cdots D_k R_2^{rc} \phi_{4j}(S_j)^{rc}, 1 \leq j \leq k$$

$$\underline{\phi_3(S_j)} \underbrace{JJ \cdots J}_{10k-t+v-2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}, 1 \leq j \leq k$$

(Assuming that the state S_j wants to write σ_v at the current head position)

$$\overline{\phi_{1x}(\sigma_i)} \phi_{1x}(\sigma_{i+1}) \cdots \phi_{1x}(\sigma_t) \underbrace{JJ \cdots J}_{i-1} R_1 \underbrace{JJ \cdots J}_{t-i} \phi_2(S_k) \phi_2(S_{k-1}) \cdots \phi_2(S_1)$$

$$B A_x \underbrace{JJ \cdots J}_{t+10k} F_1 F_2 \cdots F_i, 1 \leq i \leq t, 1 \leq j \leq k$$

$$\overline{F_i} F_{i+1} \cdots F_t \underbrace{JJ \cdots J}_{k+1} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}, 1 \leq i \leq t$$

$$\frac{\phi_2(S_j) \phi_2(S_{j-1}) \cdots \phi_2(S_1) B A_m \theta_1(NS_1(j)) \theta_2(NS_2(j)) \cdots \theta_t(NS_t(j))}{\underbrace{JJ \cdots J}_{10k} \underbrace{CC \cdots C}_{t+2}}, 1 \leq j \leq k$$

(The value of m is determined by S_j .)

There are 3 restriction enzymes in this system. The first enzyme E_1 recognizes a double stranded region KXR_1 , where K is the common short sequence shared by all sequences of the form $\phi_{1x}(\sigma_y)$, X can be any sequence of length tL and R_1 is a special sequence of length L . The cleavage on the strand with sequence KXR_1 is $(11k + 2t + 2)L$ bases from the restriction site R_1 and the cleavage on the complementary strand is $(11k + 2t + 1)L$ bases away from the restriction site KXR_1^{rc} . For the ease of explanation, when we describe the sequence E_1 recognizes, we only describe the location of sequence R_1 (or, equivalently, R_1^{rc}). Next, we let all sequences of the form $\phi_{4i}(S_j)$ have the same subsequence R_4 with $O(1)$ length in the end. The second enzyme E_2 recognizes R_4R_2 . The cleavage on the strand with sequence R_4R_2 is $(k + 1)L$ bases away from the restriction site and the cleavage on the complementary strand is kL bases away from the restriction site. The last three enzymes recognizes BYR_3^{rc} for any sequence Y with length L . This enzyme makes a cleavage $(k + 1)L$ bases away from the restriction site on the strand with sequence BYR_3^{rc} and make the other cleavage kL bases away from the restriction site on the complementary strand.

To test progressiveness and safety, we need to define a series of walker configurations P_i . We let P_i denote the configuration where the walker just simulates i steps of the Turing machine and all the “write” operation has been performed. We now explain how this walker is designed to operate and verify the safety properties for our walker proposal:

B.2 The First Property of Safety

We now describe how this walker moves and simulates the given Turing machine.

Step 1:

In the initial configuration, enzyme E_1 can recognize R_1 on the first DNA strand

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_t(NS_t(1)) \underbrace{JJ \cdots J}_{10k} \underbrace{CC \cdots C}_{t+2}$$

and makes a cleavage. After this, only the bottom part of this strand

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_t(NS_t(1)) \underbrace{JJ \cdots J}_{10k} \underbrace{CC \cdots C}_{i-1}$$

is anchored. The top part can float away.

Step 2:

The strand $\overline{C} \underbrace{JJ \cdots J}_{t+k+1} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$ can now attach using the complementary sequence \overline{C} and \underline{C} and we have

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_t(NS_t(1)) \underbrace{JJ \cdots J}_{10k} \underbrace{CC \cdots CC}_{i-1} \underbrace{JJ \cdots J}_{t+k+1} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$$

The enzyme E_1 can now recognize either R_1 or R_1^{rc} . If it recognizes R_1 first, the configuration goes back to the end of Step 1. If it recognizes R_1^{rc} , the strand will become

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1))$$

Step 3:

Notice that $NS_i(1)$ is the next state of the walker. We assume that $NS_i(1) = S_j$ for $j \in \{1, 2, \dots, k\}$.

The strand $\theta_i(S_j) \underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{j-1} \phi_{4i}(S_j) \underbrace{JJ \cdots J}_{10k+i-j-2} C_{ij}$ can attach to the anchored strand

produced by step 2. After the attachment, then enzyme E_1 can function again and cleave the double stranded region C_{ij} to get $\underline{C_{ij}}$.

Step 4:

The strand $\overline{C_{ij}} \underbrace{JJ \cdots J}_{2t+k-i+j+2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$ can attach to the anchored strand described in step 3

and we get

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1)) \underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{j-1} \phi_{4i}(S_j) \\ \underbrace{JJ \cdots J}_{10k+i-j-2} \underline{C_{ij}} \underbrace{JJ \cdots J}_{2t+k-i+j+2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$$

Next, the enzyme E_1 can recognize R_1 and R_1^{rc} . If the enzyme recognizes R_1 first, then situation goes back to the end of step 3. If the enzyme recognizes R_1^{rc} first, then we have

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1)) \underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{j-1} \overline{\phi_{4i}(S_j)}$$

Step 5:

The strand $\underline{\phi_{4i}(S_j)} R_2 \underbrace{JJ \cdots J}_{k-j-1} R_3 A_m^{rc} J D_1 D_2 \cdots D_j$ can now attach and enzyme E_2 can function,

resulting in

$$\square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1)) \underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{j-1} \phi_{4i}(S_j) R_2 \\ \underbrace{JJ \cdots J}_{k-j-1} R_3 A_m^{rc} J D_1 D_2 \cdots \underline{D_j}$$

Step 6:

The strand $\overline{D_j} D_{j+1} \cdots D_k R_2^{rc} \phi_{4j}(S_j)^{rc}$ can attach to the anchored strand produced by step 5 and the enzyme E_2 can recognize either $\phi_{4i}(S_j) R_2$ or $R_2^{rc} \phi_{4j}(S_j)^{rc}$. If $\phi_{4i}(S_j) R_2$ is recognized, the configuration goes back to the end of step 5. If $R_2^{rc} \phi_{4j}(S_j)^{rc}$ is recognized, then the strands become

$$\begin{aligned} & \square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1)) \underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{j-1} \phi_{4i}(S_j) R_2 \\ & \underbrace{JJ \cdots J}_{k-j-1} R_3 \overline{A_m^{rc}} \end{aligned}$$

Step 7:

Now, the current strand (which is the first strand anchored on the substrate) can attach to the next anchorage on the substrate using complementary sequences $\overline{A_m}$ and $\overline{A_m^{rc}}$. We get (assuming the next input symbol is σ_u)

$$\begin{aligned} & \square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1)) \underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{j-1} \phi_{4i}(S_j) R_2 \\ & \underbrace{JJ \cdots J}_{k-j-1} R_3 \overline{A_m^{rc}} \square_{m,u}^{rc} \end{aligned}$$

Now, enzymes E_2 and E_3 can both function and we will get the following two anchored strands regardless of the order of those two enzymes functioning:

$$\begin{aligned} & \square_{1,i} A_1 \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1)) \underbrace{JJ \cdots J}_{t-i} \overline{\phi_3(S_j)} \\ & \phi_{1m}(\sigma_1) \phi_{1m}(\sigma_2) \cdots \phi_{1m}(\sigma_t) \underbrace{JJ \cdots J}_{v-1} R_1 \underbrace{JJ \cdots J}_{t-v} \phi_2(S_k) \phi_2(S_{k-1}) \cdots \overline{\phi_2(S_j)} \end{aligned}$$

Originally, the walker attached on top of the first strand. After the step, the walker moves to the second strand and the symbol σ_v corresponding to state S_j will be written on the first strand. Steps 8-10 describe the reactions happening on the first strand and step 11 describes the attachment of the new walker.

Step 8:

Strand $\overline{\phi_3(S_j)} \underbrace{JJ \cdots J}_{10k-t+v-2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$ can attach to the first product strand mentioned in step 7.

After the attachment, enzyme E_1 can recognize $\{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$ and cleaves the strand. Hence, we get

$$\phi_{11}(\sigma_1) \phi_{11}(\sigma_2) \cdots \overline{\phi_{11}(\sigma_v)}$$

Step 9:

The strand

$$\begin{aligned} & \overline{\phi_{11}(\sigma_v)} \phi_{11}(\sigma_{v+1}) \cdots \phi_{11}(\sigma_t) \underbrace{JJ \cdots J}_{v-1} R_1 \underbrace{JJ \cdots J}_{t-v} \phi_2(S_k) \phi_2(S_{k-1}) \cdots \phi_2(S_1) B A_1 \\ & \underbrace{JJ \cdots J}_{t+10k} F_1 F_2 \cdots F_v \end{aligned}$$

can now attach to the anchored strand produced by step 8 and enzyme E_1 can function again to cleave F_v .

Step 10:

The strand $\overline{F_v} F_{v+1} \cdots F_t \underbrace{JJ \cdots J}_{k+1} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$ can attach to the anchored strand on step 9.

After the attachment, the enzyme E_1 can recognize either R_1 or R_1^{rc} . Similar to the previous cases, if the enzyme recognizes R_1 , the situation goes back to the end of step 9. If the enzyme recognizes R_1^{rc} , then this strand become

$$\phi_{11}(\sigma_1) \phi_{11}(\sigma_2) \cdots \phi_{11}(\sigma_t) \underbrace{JJ \cdots J}_{v-1} R_1 \underbrace{JJ \cdots J}_{t-v} \phi_2(S_k) \phi_2(S_{k-1}) \cdots \phi_2(S_1) B \overline{A_1}$$

which is just

$$\square_{1, v} \overline{A_1}$$

This shows that the walker has been removed from the first strand and the next symbol σ_v has been written on the first strand.

Step 11:

The strand

$$\underbrace{\phi_2(S_j)} \phi_2(S_{j-1}) \cdots \phi_2(S_1) B A_m \theta_1(NS_1(j)) \theta_2(NS_2(j)) \cdots \theta_t(NS_t(j)) \underbrace{JJ \cdots J}_{10k} \underbrace{CC \cdots C}_{t+2}$$

can attach to the second strand described in step 7. After this attachment, the walker with the transition table corresponding to the state S_j is attached on the next strand, thus completes one cycle of the walker movement.

The above process will repeat autonomously and the walker will move on the substrate in the same way as the Turing machine head moves.

The following strands will be generated during the walker operations:

Step 1:

$$\overline{C} \underbrace{CC \cdots C}_i, \quad 2 \leq i \leq t+1$$

Step 2:

$$\underbrace{\theta_i(NS(i))} \theta_{i+1}(NS(i+1)) \cdots \theta_t(NS(t)) \underbrace{JJ \cdots J}_{10k} \underbrace{CC \cdots C}_i \underbrace{JJ \cdots J}_{t+k+1} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}, \quad 1 \leq i \leq t$$

Step 3:

$$\overline{C_{ij}}, \quad 1 \leq i \leq t, 1 \leq j \leq k$$

Step 4:

$$\underbrace{\phi_{4i}(S_j)}_{10k+i-j-2} \underbrace{JJ \cdots J}_{2t+k-i+j+2} C_{ij} \underbrace{JJ \cdots J}_t \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}, \quad 1 \leq i \leq t, 1 \leq j \leq k$$

Step 5:

$$\overline{D_j}, \quad 1 \leq j \leq k$$

Step 6:

$$\underline{A_m^{rc}} J D_1 D_2 \cdots D_k R_2^{rc} \phi_{4j}(S_j), \quad 1 \leq j \leq k$$

Step 7:

$$\frac{\phi_3(S_j)}{\phi_2(S_1)^{rc} \phi_2(S_2)^{rc} \cdots \phi_2(S_j)^{rc}} \underbrace{JJ \cdots J}_{j-1} \phi_4(S_j) R_2 \underbrace{JJ \cdots J}_{k-j-1} R_3 \underline{A_m^{rc}} B^{rc}$$

Step 8:

$$\frac{\phi_{1x}(\sigma_v) \phi_{1x}(\sigma_{v+1}) \cdots \phi_{1x}(\sigma_t)}{B A_x \theta_1(NS_1(1)) \theta_2(NS_2(1)) \cdots \theta_i(NS_i(1))} \underbrace{JJ \cdots J}_{u-1} R_1 \underbrace{JJ \cdots J}_{t-u} \phi_2(S_k) \phi_2(S_{k-1}) \cdots \phi_2(S_1)$$

$$\underbrace{JJ \cdots J}_{t-i} \phi_3(S_j) \underbrace{JJ \cdots J}_{10k-t+v-2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc},$$

$1 \leq j \leq k$, machine writes symbol σ_v at state S_j , $x \in \{1, 2, 3\}$

Step 9:

$$\overline{F_i}, \quad 1 \leq i \leq t$$

Step 10:

$$\underline{A_m} \underbrace{JJ \cdots J}_{t+10k} F_1 F_2 \cdots F_t \underbrace{JJ \cdots J}_{k-2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$$

B.3 The Second Property for Safety

It is easy to see that in all molecules that have two complementary strands, those two strands are held by more than $2L$ base pairs. Hence, the adversary cannot break any of them to generate new sequences. Also, by our initial specification of the sequences, we know the adversary cannot stick two different strands together. Hence, the adversary can only perform the operations that can normally happen. We only need to check that in each step of the walker operation, the adversary cannot attach some unwanted sequence and block the normal walker operation forever. For this purpose, we check the walker strand and the idling strand (the anchored strands which do not have the walker on top) separately.

For an idling strand

$$\square w, j \overline{A_w}$$

the adversary can only attach the molecule generated by step 10

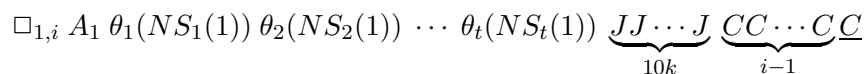
$$\underline{A_w} \underbrace{JJ \cdots J}_{t+10k} F_1 F_2 \cdots F_t \underbrace{JJ \cdots J}_{k-2} \{R_1 \underbrace{JJ \cdots J}_t K\}^{rc}$$

After the attachment, the enzyme E_1 can recognize both R_1 inside $\square w, j$ or R_1^{rc} . If the enzyme recognizes R_1^{rc} , the strand goes back to its normal form. if the enzyme recognizes R_1 , then it will cleave the double stranded region F_j and reveal $\underline{F_j}$. The configuration is now identical to the configuration at the end of step 9 and no other reactions can happen except the strand just got removed attach back (notice that there

are many copies of that strand in the solution). Therefore, no matter which reactions the adversary specifies, the strand can always go back to its original configuration using at most 2 normal reactions.

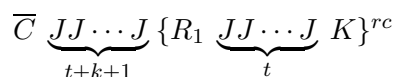
For the walker strand, we need to verify that the adversary cannot do unexpected movements at any step. We first verify this statement for the state at the end of step 1.

At the end of step 1, we have the strand

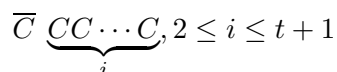


anchored on the substrate.

We want to attach the strand



to it, but the adversary can attach any strand in the following group



However, after one of these strands attaches, no further reactions can happen until it gets cleaved by the enzyme again and goes back to the situation at the end of step 1.

The situation for all other steps are the same. Starting from the end of step i , the adversary can only bring the configuration back to the beginning of step i . The walker strand can always go back to the normal state after 2 reactions.

Therefore, we know that no matter what the adversary does, the walker strand and the two adjacent anchored strands can go back to their desired state after a constant number of operations. All other strands can also go back to their normal configuration within a constant number of normal operations. This shows the second property for safety is satisfied.

B.4 Strong Safety

For strong safety, first notice that in this construction, any adversary operation must also be a normal operation and when an enzyme makes a cut, the total length of the sticky ends is exactly c . Therefore, the second property for strong safety is true. Second, at each anchorage, it is easy to see that only a constant number of each type of adversary operations are available, hence the third property is also true. Finally the ligation in step 11 is irreversible with adversary operations. From the above discussion, we conclude that the walker will perform the designed operation unless some very rare event happens.

B.5 Low Energy Consumption

Similar to our DFA walker construction, we can see that no two molecules in the solution can react with each other. Since anchorages of the form $\square_{w,j} \overline{A_w}$ do not react with any molecule in the solution, we know that all of the reactions either happen at the anchorage with the walker on top or is doing the “write” operation (steps 8-10 in the walker cycle). Also, when the walker moves from anchorage i to $i+1$, we know that the i -th anchorage will go back to the form $\square_{w,j} \overline{A_w}$ within an expected constant number of reactions in the safety proof. Therefore, the total number of operations in the “write” operation is linear in the Turing machine steps. On the other hand, at any given time, the number of operations that can happen on the anchorage with

the walker on top is at most a constant. Therefore, the number of reactions that happens on the walker is also linear in the number of operations for the Turing machine simulated. Combine the above two, we know that the total amount of energy required is linear in the number of Turing machine steps simulated.