Autonomous Programmable Nanorobotic Devices Using DNAzymes

John H. Reif and Sudheer Sahu

Department of Computer Science, Duke University
Box 90129, Durham, NC 27708-0129, USA
{reif,sudheer}@cs.duke.edu

Abstract. A major challenge in nanoscience is the design of synthetic molecular devices that run *autonomously* and are *programmable*. DNA-based synthetic molecular devices have the advantage of being relatively simple to design and engineer, due to the predictable secondary structure of DNA nanostructures and the well-established biochemistry used to manipulate DNA nanostructures. We present the design of a class of DNAzyme based molecular devices that are autonomous, programmable, and further require no protein enzymes. The basic principle involved is inspired by a simple but ingenious molecular device due to Mao et al [25]. Our DNAzyme based designs include (1) a finite state automata device, *DNAzyme FSA* that executes finite state transitions using DNAzymes, (2) extensions to it including probabilistic automata and non-deterministic automata, (3) its application as a *DNAzyme router* for programmable routing of nanostructures on a 2D DNA addressable lattice, and (4) a medical-related application, *DNAzyme doctor* that provide transduction of nucleic acid expression: it can be programmed to respond to the underexpression or overexpression of various strands of RNA, with a response by release of an RNA.

1 Introduction

1.1 Prior Autonomous Molecular Computing Devices

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

(a) **DNA Tiling Assemblies.** One theoretical concept that had considerable impact on experimental demonstrations was that of Wang Tiling. This is an abstract model that allows for a finite set of 2D rectangles with labeled sides to assemble 2D lattices by appending together tiles at their matching sides. Winfree first proposed the use of DNA nanostructures known as *DNA tiles* to achieve universal computations. DNA tiles self-assemble into 2D lattices as determined by the tiles' pads (ssDNA on the sides of the tiles that can hybridize to other tiles' pads). The last decade has seen major successes in experimental demonstrations of the use of such DNA tiling assemblies to construct patterned lattices and tiling computations. DNA tiling assemblies have been used effectively in construction of periodic two-dimensional lattices, such as those made from double-crossover (DX) DNA tiles [29], rhombus tiles [12], triple-crossover (TX)

M.H. Garzon and H. Yan (Eds.): DNA 13, LNCS 4848, pp. 66–78, 2008.

tiles [9], and "4x4" tiles [31], as well as triangle lattices [11] and hexagonal lattices [5]. They have also been used for the construction of patterned lattices [30] by designing the DNA tile pads to program computations. The use of DNA tiling assembly has two major advantages over most other methods for molecular computation, since it: (i) operates entirely autonomously, without outside mediated changes, and (ii) does not require the use of protein enzymes.

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

(b) **Autonomous Molecular Computing Devices that Execute Multiple State Transitions.** There are only two other known methods for DNA computation that operate autonomously. Both use ingenious constructions, but require the use of enzymes.

(i) The *whiplash PCR machines* of [14,15,19,28]. These however, can only execute a small number of steps before they require changes in the environment to execute further steps. Also, they require the use of polymerase enzyme.

(ii) The autonomous DNA machines of Shapiro[4,2,3], which execute finite transitions using restriction enzymes. The autonomous DNA machine [3] demonstrated molecular sensing and finite state response capabilities for that could be used for medical applications (though the demonstrations were made in test tubes only, rather than in natural biological environments as would be required for their medical applications). Their paper was important motivational factor in the work described here.

1.2 Our Main Contribution

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

1.3 DNA Nanomechanical Devices

Prior Nonautonoumous Nanomechanical DNA Devices. A variety of DNA nanomechanical devices have been constructed that exhibit motions such as open/close [23,24,34], extension/contraction [1,8,10], and rotation [13,26,32]. The motion of these devices is mediated by external environmental changes such as the addition and removal of DNA fuel strands [1,8,10,23,24,26,32,34] or the change of ionic strength of the solution [13]. For example, non-autonomous progressive walking devices, mediated by the addition and removal of DNA strands, were constructed both by Seeman [21] and Pierce [22]. Although in many cases ingeniously designed, these devices need external (human or automation-based) intervention for each step of their motions. These synthetic DNA devices are in sharp contrast with cellular protein motors and machines on macroscale that operate autonomously, without requiring any interference.

Recent times have seen significant progress in construction of DNA nanomechanical devices that execute autonomous, progressive motions. Reif [17] gave two designs for autonomous DNA nanomechanical devices that traverse bidirectionally along a DNA nanostructure. Turberfield et al proposed using DNA hybridization energy to fuel autonomous free-running DNA machines [27]. Peng et al [33] was the first to experimentally demonstrate an autonomous DNA walker, which is an autonomous DNA device in which a DNA fragment translocates unidirectionally along a DNA nanostructure. It used DNA ligase and restriction enzymes.

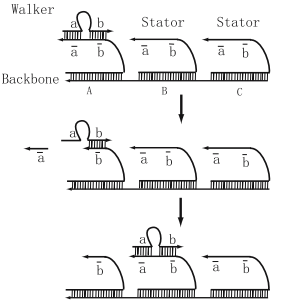

Fig. 1. Overview of Mao's crawler [25] constructed using DNA enzyme

Recently Mao demonstrated two autonomous DNA nanomechanical devices driven by DNA enzymes (non-protein), namely (a) a tweezer [7,6] which is a DNA nanostructure that open and closes autonomously and (b) a DNA crawler [25] using DNA enzyme (DNAzyme), which traverses across a DNA nanostructure.

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

1.4 Overview of This Paper and Results

The goal of this paper is to address the above limitations, providing DNAzyme based devices with substantially enhanced functionalities. We present the design of *DNAzyme FSA*: a finite state machine based on the activity of DNAzyme and strand displacements in Section 2. DNAzyme FSA can be easily extended to non-deterministic finite state automata and probabilistic automata as described in Section 2.6. In Section 3 we present a medical related application of DNAzyme FSA referred to as *DNAzyme doctor*. DNAzyme doctor is a molecular computer for logical control of RNA expression using DNAzyme. Another application of DNAzyme FSA, *DNAzyme router*: a DNAzyme

based system for programmable routing of the walker on a 2D lattice is described in Section 4. All the devices described in this paper are based on selective cleaving activity of DNAzyme and strand displacement processes.

2 DNAzyme FSA: DNAzyme Based Finite State Automata

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

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

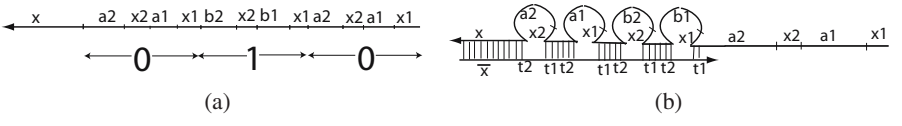

Fig. 2. (a) Encoding of 0 and 1 in DNAzyme FSA. (b) Protector strand partially hybridizes with the input strand to form bulge loops. The sticky end formed at the end of the input strand outside of the bulge loops represents the active input symbol. This scheme protects the input symbols other than the currently active symbol from becoming active.

2.1 Encoding the Input Symbols

First of all, we describe the way the input is encoded for the DNAzyme FSA. Input symbols 0 and 1 are encoded as the RNA sequences $x_1 \cdot a_1 \cdot x_2 \cdot a_2$ and $x_1 \cdot b_1 \cdot x_2 \cdot b_2$, respectively, where a_1, a_2, b_1, b_2, x_1, and x_2 are RNA sequences, and $\cdot$ represents concatenation. Figure 2 (a) illustrates this encoding of the input symbols. It should be noted that 0 and 1 share common subsequences x_1 and x_2. Also, there is a special subsequence x at the end of the input subsequence. This is central to the working of the DNAzyme FSA as will be explained later.

2.2 Active Input Symbol

While encoding the input for DNAzyme FSA, it is essential to have a mechanism to detect the current input symbol that is being scanned by DNAzyme FSA. We will refer to this symbol as *active input symbol*. In order to implement this feature in DNAzyme FSA only a small segment of the RNA strand encoding the input symbols is kept active. Most part of it is kept protected by hybridization with a partially complementary sequence, referred to as *protecting sequence*. It has not been shown in the figure but

the protecting sequence should not be one continuous strand. Instead it should contain nicks at various positions. This is necessary for the working of device and will be explained later. The active input symbol is represented by the sticky end of the RNA sequence encoding the input. We refer to this nanostructure as *input nanostructure*. Figure 2 (b) illustrates the idea. The input nanostructure encodes the input 010. The active input symbol is rightmost 0 (in 010), and it is encoded by the sticky end of the input nanostructure, and hence is active. However, the leftmost 0 and the 1 are encoded in the protected portion of the input nanostructure. They have been protected by hybridization with a protecting sequence. Since the protecting sequence is partially complementary to the sequence encoding the input symbols, it results in the formation of bulge loops. In the Figure 2 b) a_2, a_1, b_2, and b_1 contain a subsequence complementary to t_2, while x_2 and x_1 contain subsequence complementary to t_1. Since the RNA sequence encoding input is partially complementary to the protecting sequence $t_2.t_1.t_2.t_1...$ it forms the bulge loop structure as shown in the Figure 2 (b). Each input symbol is hence represented by two bulge loops. It should be noted that the special sequence x at the end of the input sequence and $\bar{x}$ at the end of protecting sequence ensure that only the desired alignment of protecting sequence with input sequence is favored. As a result, only the desired input nanostructure as shown in Figure 2 (b) is formed.

2.3 States and Transitions

After the description of the input, next we describe the design of states and transitions in finite state machine. In DNAzyme FSA, a network of DNAzymes is embedded on a two-dimensional plane, and the input nanostructure is routed over it. The state of the DNAzyme FSA at any time is indicated by the DNAzyme that holds the input nanostructure at that time. During each state transition of DNAzyme FSA, the segment of input nanostructure encoding the active input symbol is cleaved, the next bulge loop opens up exposing the segment encoding next input symbol, thereby making it new active input symbol, and the input nanostructure jumps to another DNAzyme that indicates the new state of DNAzyme FSA. In subsequent paragraphs, we will explain in details the complete process of state transition in DNAzyme FSA. As shown in Figure 3 (a), a state transition from one state to another is implemented as two evenly spaced DNAzymes, referred to as *transition machinery* for that state transition. Each of these DNAzymes is tethered to another DNA nanostructure, which forms part of the backbone of the DNAzyme FSA. DNAzyme D_{0,s_1} and D'_{0,s_2} form the transition machinery for state transition from state s_1 to state s_2 for input 0. Similarly, DNAzyme D_{1,s_1} and D'_{1,s_2} form the transition machinery for state transition from state s_1 to state s_2 for input 1. It should be noted that in our nomenclature the first subscript of the DNAzyme specifies the active input symbol and the second subscript specifies the states for a transition machinery.

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

In the transition machinery for state transition for input 0, the DNAzymes D_{0,s_1} and D'_{0,s_2} contain DNA subsequences $\overline{x_2} \cdot \overline{a_1} \cdot \overline{x_1}$ and $\overline{x_1} \cdot \overline{a_2} \cdot \overline{x_2}$ respectively, at their free ends. The DNA subsequences of D_{0,s_1} is partially complementary to the RNA sequence

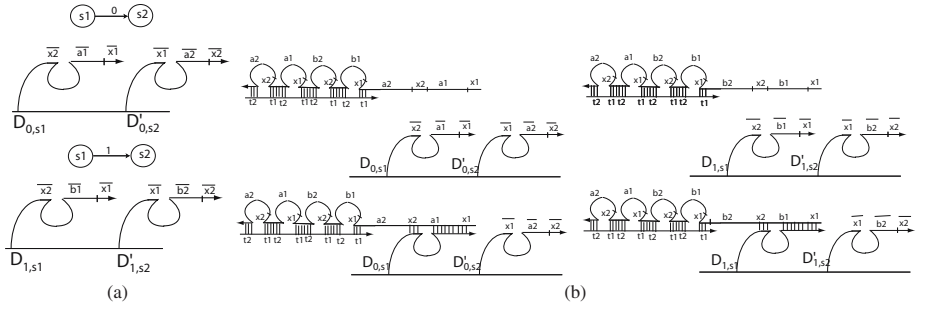

Fig. 3. (a) Figure illustrates the implementation of a state transition through DNAzymes. (b) D_{0,s_1} in the transition machinery for state transition at 0 combines with input nanostructure when active input symbol encoded by the sticky end is 0. When the active input symbol encoded by the sticky end is 1, D_{1,s_1} in the transition machinery for state transition at 1 combines with the input nanostructure.

that encode the symbol 0 ($x_1 \cdot a_1 \cdot x_2 \cdot a_2$). This ensures that only when the sticky end of input nanostructure is $x_1 \cdot a_1 \cdot x_2 \cdot a_2$, it can hybridize with the DNAzyme D_{0,s_1}. Thus a state transition for 0 is not taken in DNAzyme FSA, unless the active input symbol is 0.

Similarly, in the transition machinery for state transition for input 1, the DNAzymes D_{1,s_1} and D'_{1,s_2} contain DNA subsequences $\overline{x_2} \cdot \overline{b_1} \cdot \overline{x_1}$ and $\overline{x_1} \cdot \overline{b_2} \cdot \overline{x_2}$ respectively, at their free ends. These subsequences are partially complementary to the RNA sequence that encode the symbol 1 ($x_1 \cdot b_1 \cdot x_2 \cdot b_2$). As explained earlier, this ensures that a state transition for 1 is not taken in the DNAzyme FSA, unless the active input symbol is 1. Figure 3 (b) further illustrates the idea.

2.4 Description of State Transition

In this section, we will describe the movement of the input nanostructure over the DNAzymes in a transition machinery to carry out the state transition in DNAzyme FSA. Figure 4 (a) shows a transition machinery for input 0. Initially, the input nanostructure is hybridized with the DNAzyme D_{0,s_1}. The sticky end of the input nanostructure represents the active input symbol 0, and therefore, the transition at input 0 is to be performed. First, the DNAzyme D_{0,s_1} cleaves the input nanostructure as shown in Figure 4 (a). Now the sticky end of input nanostructure has only x_2 as complementary subsequence to the subsequence $\overline{x_2} \cdot \overline{a_1} \cdot \overline{x_1}$ at the free end of DNAzyme D_{0,s_1}. However, the longer subsequence $x_2 \cdot a_2$ in its sticky end is complementary with the subsequence $\overline{a_2} \cdot \overline{x_2}$ of DNAzyme D'_{0,s_2}. Therefore, a strand displacement process takes place with the free ends of DNAzymes D_{0,s_1} and D'_{0,s_2} competing against each other to hybridize with sticky end ($x_2 \cdot a_2$) of the input nanostructure. Since D'_{0,s_2} provides a longer complementary subsequence, ultimately D_{0,s_1} is displaced and the input nanostructure is now hybridized with D'_{0,s_2} as shown in Figure 4 (a). It should be noted that the next bulge loop gets opened in this process. An input symbol is encoded across two bulge

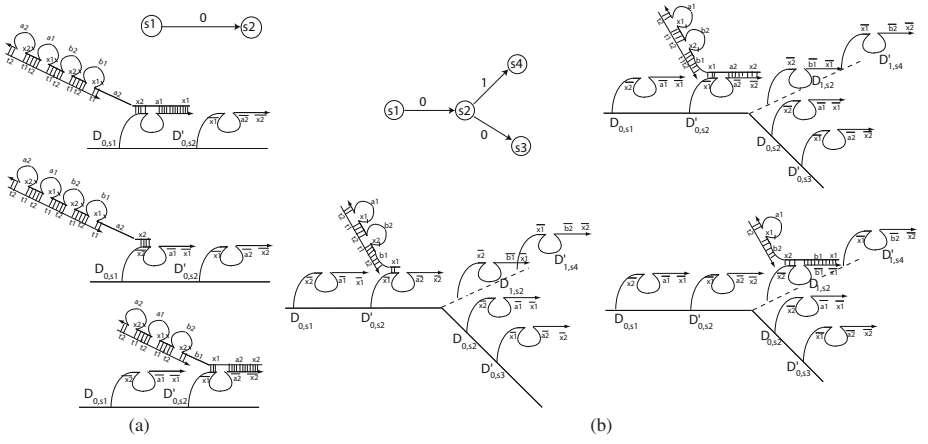

Fig. 4. (a) First half of a state transition by DNAzyme FSA from s_1 to s_2 at input 0 is illustrated. Sequence encoding active input symbol 0 gets cleaved by DNAzyme D_{0,s_1}, input nanostructure moves to next DNAzyme D'_{0,s_2} by strand displacement, and the next bulge loop in the input nanostructure opens up in the process. (b) Second half of a state transition by DNAzyme FSA from s_1 to s_2 at input 0 is shown. The mechanism is similar to the first half. However, in this part the next input symbol and next state transition of DNAzyme FSA is determined, and the input nanostructure lands up on the appropriate transition machinery for the next state transition to begin correctly.

loops in the input nanostructure. As the first half of the sticky end $(x_1 \cdot a_1)$ encoding the half of the active input symbol 0 got cleaved, the current sticky end is $x_2 \cdot a_2 \cdot x_1 \cdot b_1$, that contains half of the sequence encoding symbol 0 and half of the sequence encoding the symbol 1. This completes the first half of the state transition by DNAzyme FSA.

The second half of the transition in DNAzyme FSA takes place in exactly similar manner. Half of the sticky end $(x_2 \cdot a_2)$ of the input nanostructure that encodes the remaining half of the active input symbol 0 gets cleaved, thus leaving only x_1 as complementary to free end of DNAzyme D'_{0,s_2} $(\overline{x_1} \cdot \overline{a_2} \cdot \overline{x_2})$. At this point the sticky end of the input nanostructure is $x_1 \cdot b_1$ which is half of the sequence that encodes the input symbol 1. It indicates that the next active input symbol is 1 and therefore, the next state transition should be from state s_2 at input 1. This is ensured by the DNAzyme FSA in the following way. Since the sticky end of the input nanostructure is $(x_1 \cdot b_1)$, the DNAzyme D_{1,s_2} that has the sequence $\overline{x_2} \cdot \overline{b_1} \cdot \overline{x_1}$ at its free end gets involved in strand displacement with D'_{0,s_2} to hybridize with the sticky end $(x_1 \cdot b_1)$ of input nanostructure. Because of the longer complementary sequence D_{1,s_2} ultimately displaces D'_{0,s_2} and hybridizes with the sticky end of nanostructure. This results in the opening of next bulge loop in input nanostructure as shown in Figure 4 (b).

It should be noted that D_{0,s_2} (with sequence $\overline{x_1} \cdot \overline{b_2} \cdot \overline{x_2}$ at its free end) does not have sequences complementary to the sticky end $(x_1 \cdot b_1)$ of input nanostructure, so it can not get involved in any strand displacement. Therefore, the input nanostructure is guaranteed to move to the DNAzyme D_{1,s_2}. After the opening of the next bulge loop,

the new sticky end ($x_1 \cdot b_1 \cdot x_2 \cdot b_2$) of input nanostructure encodes the input symbol 1. Thus, the input nanostructure lands up in the appropriate transition machinery for the next state transition, and the next state transition at input 1 can begin correctly.

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

Figure 4 (b) illustrates the second half of the state transition of DNAzyme FSA.

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

2.5 Complete State Machine

The components described above can be integrated to implement the complete finite state automata. Any state transition in the DNAzyme FSA can be implemented by two DNAzymes as described earlier. These DNAzymes are embedded on a nanostructure that forms the backbone of the DNAzyme FSA. The addressable nanostructures formed by DNA origami [20] or fully-addressable DNA tile lattices [16] might provide useful nanostructures for this backbone. Hence, the state machine can be laid out on this nanostructure by implanting a network of DNAzymes on it. The input nanostructure traverses over them in a programmable way and keeps getting cleaved in the process.

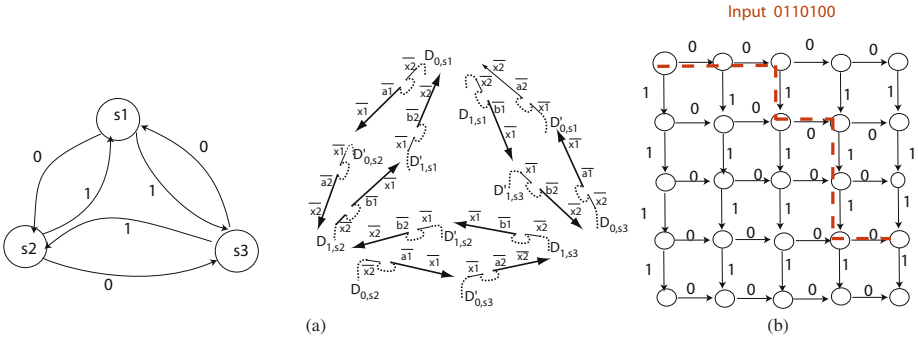

Fig. 5. (a) The DNAzyme implementation of the finite state machine shown on left. (b) Illustration of programmable routing in two dimensions.

Figure 5 (a) shows an implementation of a DNAzyme FSA (at the right) for the finite state automata (at the left). It should be noted that the DNAzymes shown in the Figure 5 (a) are actually implanted on a backbone nanostructure. The dashed lines represent the sides of these DNAzymes that are embedded in the backbone nanostructure.

The output of the DNAzyme FSA is detected using insitu hybridization techniques. The details of the protocol are described in [18].

2.6 Non-deterministic and Probabilistic DNAzyme FSA

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

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

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

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

3 DNAzyme Doctor: A Molecular Computer for Logical Control of RNA Expression Using DNAzyme

The finite state automaton described in Section 2 can be used in various computational and routing applications. In this section we describe DNAzyme doctor, an application related to medical field. It is an autonomous molecular computer for control of RNA expression based on the overexpression and underexpression of other RNAs. Earlier Shapiro[3] had constructed a molecular computer using protein enzymes for logical control of RNA expression. DNAzyme doctor performs the same function, while completely eliminating the use of protein enzymes in the design. For the ease of illustration let us consider a similar example as given in [3]. Suppose a disease is diagnosed positive if RNAs R_1 is underexpressed, R_2 is underexpressed, R_3 is overexpressed, and R_4 is overexpressed. Thus, the detection of the disease can be done by computing logical AND of the above mentioned four RNA expression tests. In case it is established that the disease exists, a curing drug should be released. While in any other case, the drug should not be released. Figure 6 (a) illustrates the aforementioned logic in the form of a state diagram.

The sequences y_1, y_2, y_3 and y_4 are characteristic sequences of RNAs R_1, R_2, R_3, and R_4 respectively. If R_1 is overexpressed then y_1 is in excess, and if R_2 is overexpressed

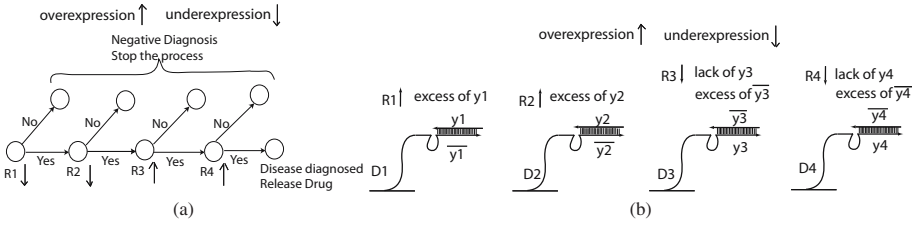

Fig. 6. (a)A state diagram for DNAzyme doctor that controls the release of a drug RNA on the basis of the RNA expression tests for the a disease (b) The figure shows the consequences of overexpression and underexpression of different RNAs on the concentrations of the respective characteristic sequences. The overexpression of R_1 and R_2 results in excess of y_1 and y_2 respectively, and they block the path of input nanostructure by hybridizing with D_1 and D_2. Similarly underexpression of R_3 and R_4 results in excess of $\overline{y_3}$ and $\overline{y_4}$ respectively, to block the path of input nanostructure.

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

Since the DNAzyme doctor only needs to perform a logical AND, it can be implemented in a simple way. We make the input nanostructure walk over four DNAzyme stators implanted on a nanostructure in a straight path (more details in [18]). Each DNAzyme stator represents one of the RNA expression test. In case the test is positive, the input nanostructure moves to next DNAzyme stator, otherwise it gets stuck and ultimately floats away in the solution. Therefore, the successful traversal of input nanostructure over all these DNAzyme stators implies that all tests are positive, and hence positive diagnosis of the disease.

In case the first test is negative (ie. overexpression of R_1), then excessively floating y_1 can bind to $\overline{y_1}$ part of the DNAzyme D_1. Similarly if second, third, or fourth tests are negative (ie.. overexpression of R_2, underexpression of R_3 or underexpression of R_4), then excessively floating y_2, y_3, or y_4 can bind to $\overline{y_2}$, $\overline{y_3}$, $\overline{y_4}$ portions of DNAzyme D_2, D_3, or D_4, respectively. The principle idea is illustrated in Figure 6. More details of DNAzyme doctor are presented in [18].

4 DNAzyme Router

For any arbitrary path along the network of DNAzymes in a given DNAzyme FSA, an input nanostructure can be designed to traverse along that path. This principle can be used for the design of a programmable routing system. The input nanostructure that moves over the DNAzyme FSA is referred to as *walker* and the complete system as DNAzyme router. The path of the walker is programmed through the state transitions of the automata and the input symbols encoded in the walker. As an example, we can create a state machine on a rectangular grid (Figure 5 (b)), in which you move right if the input is 0, and towards bottom if the input is 1. Then an input nanostructure that

represents the input 0110100 can be made to walk through the path shown by dashed lines in Figure 5 (b).

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

5 Conclusion

We have described the construction of various devices based on the DNAzymes. DNAzymes evolve through invitro selection procedures, and these processes can be designed to generate DNAzymes that cut distinct sequences. In the DNAzyme FSA, the number of DNAzymes required is proportional to the number of transitions in the automata. For binary-coded inputs the number of transitions is proportional to number of states. It should be noted that each of the devices described in the paper need the DNAzymes to be mounted on an addressable two-dimensional nanostructure such as the ones constructed by Rothemund [20] or Park et al [16], which themselves are floating in the solution. The molecular computer for logical control of RNA expression can be useful in medical field if it can be used inside a cell, and the programmable walkers can be a really useful tool in nanopartical transportation systems at nanoscale. In conclusion, the designs provided in this paper might provide useful insight for research into many interesting problems in nanotechnology.

Acknowledgement. The work is supported by NSF EMT Grants CCF-0523555 and CCF-0432038.

References

1. Alberti, P., Mergny, J.: DNA duplex-quadruplex exchange as the basis for a nanomolecular machine. Proc. Natl. Acad. Sci. USA 100, 1569–1573 (2003)
2. Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, Z., Shapiro, E.: DNA molecule provides a computing machine with both data and fuel. Proc. Natl. Acad. Sci. USA 100, 2191–2196 (2003)
3. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expression. Nature 429, 423–429 (2004)
4. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules. Nature 414, 430–434 (2001)
5. Chelyapov, N., Brun, Y., Gopalkrishnan, M., Reishus, D., Shaw, B., Adleman, L.: DNA triangles and self-assembled hexagonal tilings. J. Am. Chem. Soc. 126, 13924–13925 (2004)
6. Chen, Y., Mao, C.: Putting a brake on an autonomous DNA nanomotor. J. Am. Chem. Soc. 126, 8626–8627 (2004)
7. Chen, Y., Wang, M., Mao, C.: An autonomous DNA nanomotor powered by a DNA enzyme. Angew. Chem. Int. Ed. 43, 3554–3557 (2004)
8. Feng, L., Park, S., Reif, J., Yan, H.: A two-state DNA lattice switched by DNA nanoactuator. Angew. Chem. Int. Ed. 42, 4342–4346 (2003)

9. LaBean, T., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J., Seeman, N.: The construction, analysis, ligation and self-assembly of DNA triple crossover complexes. J. Am. Chem. Soc. 122, 1848–1860 (2000)
10. Li, J., Tan, W.: A single DNA molecule nanomotor. Nano Lett. 2, 315–318 (2002)
11. Liu, D., Wang, M., Deng, Z., Walulu, R., Mao, C.: Tensegrity: Construction of rigid DNA triangles with flexible four-arm dna junctions. J. Am. Chem. Soc. 126, 2324–2325 (2004)
12. Mao, C., Sun, W., Seeman, N.: Designed two-dimensional DNA holliday junction arrays visualized by atomic force microscopy. J. Am. Chem. Soc. 121, 5437–5443 (1999)
13. Mao, C., Sun, W., Shen, Z., Seeman, N.: A DNA nanomechanical device based on the B-Z transition. Nature 397, 144–146 (1999)
14. Matsuda, D., Yamamura, M.: Cascading whiplash pcr with a nicking enzyme. In: Hagiya, M., Ohuchi, A. (eds.) DNA Computing. LNCS, vol. 2568, pp. 38–46. Springer, Heidelberg (2003)
15. Nishikawa, A., Hagiya, M.: Towards a system for simulating DNA computing with whiplash PCR. In: Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.) Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 960–966, 6–9. IEEE Press, Washington (1999)
16. Park, S.H., Pistol, C., Ahn, S.J., Reif, J.H., Lebeck, A.R., Dwyer, C., LaBean, T.H.: Finite-size, fully addressable dna tile lattices formed by hierarchical assembly procedures. Angew. Chem. Int. Ed. 45, 735–739 (2006)
17. Reif, J.: The design of autonomous DNA nanomechanical devices: Walking and rolling DNA. In: Hagiya, M., Ohuchi, A. (eds.) DNA Computing. LNCS, vol. 2568, pp. 22–37. Springer, Heidelberg (2003), Published in Natural Computing, DNA8 special issue, vol. 2, pp. 439–461 (2003)
18. Reif, J.H., Sahu, S.: Autonomous programmable dna nanorobotic devices using dnazymes. Technical Report CS-2007-06, Duke University, Computer Science Department (2007)
19. Rose, J.A., Deaton, R.J., Hagiya, M., Suyama, A.: Pna-mediated whiplash pcr. In: Jonoska, N., Seeman, N.C. (eds.) DNA Computing. LNCS, vol. 2340, pp. 104–116. Springer, Heidelberg (2002)
20. Rothemund, P.: Generation of arbitrary nanoscale shapes and patterns by scaffolded DNA origami. Nature (2005)
21. Sherman, W., Seeman, N.: A precisely controlled DNA biped walking device. Nano Lett. 4, 1203–1207 (2004)
22. Shin, J., Pierce, N.: A synthetic DNA walker for molecular transport. J. Am. Chem. Soc. 126, 10834–10835 (2004)
23. Simmel, F., Yurke, B.: Using DNA to construct and power a nanoactuator. Phys. Rev. E 63, 41913 (2001)
24. Simmel, F., Yurke, B.: A DNA-based molecular device switchable between three distinct mechanical states. Appl. Phys. Lett. 80, 883–885 (2002)
25. Tian, Y., He, Y., Chen, Y., Yin, P., Mao, C.: Molecular devices - a DNAzyme that walks processively and autonomously along a one-dimensional track. Angew. Chem. Intl. Ed. 44, 4355–4358 (2005)
26. Tian, Y., Mao, C.: Molecular gears: A pair of DNA circles continously rolls against each other. J. Am. Chem. Soc. 126, 11410–11411 (2004)
27. Turberfield, A., Mitchell, J., Yurke, B., Mills, J.A.P., Blakey, M., Simmel, F.: DNA fuel for free-running nanomachines. Phys. Rev. Lett. 90, 118102 (2003)
28. Winfree, E.: Whiplash pcr for o(1) computing. Technical Report 1998.23, Caltech (1998)
29. Winfree, E., Liu, F., Wenzler, L., Seeman, N.: Design and self-assembly of two-dimensional DNA crystals. Nature 394(6693), 539–544 (1998)
30. Yan, H., LaBean, T., Feng, L., Reif, J.: Directed nucleation assembly of DNA tile complexes for barcode patterned DNA lattices. Proc. Natl. Acad. Sci. USA 100(14), 8103–8108 (2003)

31. Yan, H., Park, S., Finkelstein, G., Reif, J., LaBean, T.: DNA-templated self-assembly of protein arrays and highly conductive nanowires. Science 301(5641), 1882–1884 (2003)
32. Yan, H., Zhang, X., Shen, Z., Seeman, N.: A robust DNA mechanical device controlled by hybridization topology. Nature 415, 62–65 (2002)
33. Yin, P., Yan, H., Daniell, X., Turberfield, A., Reif, J.: A unidirectional DNA walker moving autonomously along a linear track. Angew. Chem. Int. Ed. 43, 4906–4911 (2004)
34. Yurke, B., Turberfield, A., Mills, J.A.P., Simmel, F., Neumann, J.: A DNA-fuelled molecular machine made of DNA. Nature 406, 605–608 (2000)