International Journal of Computer Science & Applications
Vol. III, No. I, pp. 51 -60

A New DNA Implementation of Finite State Machines*

A. Nowzari-Dalini[†,1,2], E. Elahi[2,3,4], H. Ahrabian[1,2], M. Ronaghi[5]

[1] *Center of Excellent of Biomathematics,*
School of Mathematics, Statistics, and Computer Science,
University of Tehran, Tehran, Iran.

[2] *Department of Bioinformatics,*
Institute of Biochemistry and Biophysics,
University of Tehran, Tehran, Iran.

[3] *School of Biology,*
University of Tehran, Tehran, Iran.

[4] *National Institute for Genetic Engineering and Biotechnology,*
Tehran, Iran.

[5] *Stanford Genome Technology Center,*
Stanford University, 855 California Avenue, Palo Alto, CA 94304, USA.

Abstract

Two new models for implementing finite state machines with DNA computing are presented. The operations used in both models are simple and easy to implement. Operations include immobilization of DNA strands onto paramagnetic beads, DNA hybridization, DNA ligation and restriction enzyme cleavage. Use of paramagnetic beads greatly reduces performance time and demonstrates DNA chip compatibility of the models. In one of the models, the length of DNA strands created during the intermediate operations are independent of the length of the input string. Optical extraction in both models detects the final state.

* This research was partially supported by University of Tehran.

† To whom correspondence should be addressed, E-mail: nowzari@khayam.ut.ac.ir.

1. Introduction

DNA computing is an emerging field bridging the gap between computer science and molecular genetics. Adleman's experiments [1, 4] were powerful demonstrations of DNA computing, proposing computational solutions to a simple instance of the Hamiltonian Path Problem to be achieved in a test tube and presenting the potential of DNA as an alternative device for massive parallel computation. Lipton [15] proposed an approach for using DNA to solve satisfiability and other problems in computational class NP. Ogihara and Ray [17] presented a method for simulating Boolean circuits. Even, few polynomial time algorithms such as shortest path problem in graph are solved with DNA computing [13, 21]. The power and feasibility of DNA computing is also explored by examining its application to problems of low level complexity classes. A few groups have implemented different models of the Turing machine based on DNA molecules [4, 11, 19]. They designed universal models of computation and showed the power of DNA computation. Subsequently, others [5, 6, 9] explored the potential of DNA computing by DNA implementation of finite state machines (or simply automata). Though promising and intellectually satisfying, many of the suggested protocols are tedious and not appropriate for automation. The protocol presented by Benenson and colleagues [5] is most attractive, though final detection remains labor intensive.

In this paper, two new DNA implementations of deterministic automata are resented. The objective in their design was to simplify and reduce the number of processes used in the implementations by combining solid support and enzymatic reactions. In both models, solid supports, ligation, hybridization, and restriction enzyme cleavage reactions are used, and the final state is detected optically. The second model includes an additional cutting reaction which renders the length of strands created during intermediate operations independent of the length of the input string. Both models are illustrated with an example. The models can be also used for implementing nondeterministic finite state machines.

2. Implementing Finite State Machines

A finite state machine is a device which makes computations by reading a one-way read only tape. The input consists of "words" written on the tape. The written words use a specified alphabet which is called the input alphabet and the words constitute a string. This device has a finite control which reads a problem instance on the input tape and makes appropriate computations according to a prewired program. Application of the program results in driving the machine from its current (source) state to the next (target) state depending on the symbols most recently read on the tape. At the end of the process, it becomes apparent whether the input is accepted or rejected by the machine.

Formally, a deterministic finite state automata is defined by the quintuple $M=(Q, S, d, q_0, F)$, where Q is a finite set of internal states, S is a finite set of symbols called the input alphabets, $d: Q\times S\rightarrow Q$ is a total function called the transition function, $q_0\in Q$ is the initial state, and $F\subseteq Q$ is a set of final states. For a nondeterministic finite state automata, d could be a relation, $d: Q\times(S\cup\{l\})\rightarrow 2^Q$, where l is an empty string [12]. An example of deterministic

finite state automata whose implementation will be discussed in this section is presented in Figure 1. This machine detects the divisibility of inputs in the form of a binary string by 4, and can be expressed by $M=(\{q_0,q_1,q_2\}, \{s_0,s_1\}, d, q_0, \{q_0\})$, where $s_0=0$ and $s_1=1$, and δ is given by $d(q_0,s_0)=q_0$, $d(q_0,s_1)=q_1$, $d(q_1,s_0)=q_2$, $d(q_1,s_1)=q_1$, $d(q_2,s_0))=q_0$, $d(q_2,s_1)=q_1$.

Initially, the simulation of our finite state automata begins by encoding of the input alphabets, states and transition function as a DNA strands. Later, the necessary steps for ascertaining acceptance or rejection of an input string by an automata are explained. It is assumed v be the total number of states $Q=\{q_0,\ldots,q_{v-1}\}$ in the automata and p be the total number of the elements in the alphabet $S=\{s_0,\ldots,s_{p-1}\}$. The DNA strands are presented as follows:

1) x_i $(i=0,\ldots,v-1)$, is a sequence of nucleotides of a defined length, one for each of the states (written 5' to 3'). Limitations on the nature of this sequence and other sequences are later presented..
2) $\bar{x}_I$ $(i=0,\ldots,v-1)$, is a sequences of nucleotides of a defined length, corresponding to the complement of any x_i (written 3' to 5').
3) a_{s_i} $(i=0,\ldots,p-1)$, is a sequence of nucleotides of a defined length, one for each of the elements of the input alphabet (written 5′ to 3′).
4) $\bar{a}_{s_i}$ $(i=0,\ldots,p-1)$, is a sequences of nucleotides of a defined length, corresponding to the complement of a_{s_i} (written 3′ to 5′).

Now for each transition function elements, $d(q_i,s_k)=q_j$, two DNA strands are created:

a) 5'-x_i + a_{s_k}-3',

b) 3'-$HR(\bar{x}_i)$ + $\bar{a}_{s_k}$ + $HL(\bar{x}_j)$ -5',

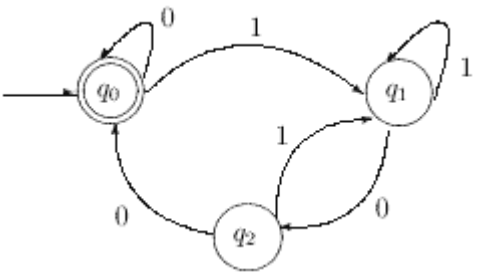


Figure 1: A finite state automata for divisibility by 4.

where $HR(\bar{x}_i)$ represents the right half part of $\bar{x}_i$ and $HL(\bar{x}_i)$ the left half part of $\bar{x}_j$, and "+" shows the joint of the strands. In addition to the above strands, the $HL(\bar{x}_0)$ strand is created for the state q_0.

As hybridization reactions are essential components in the implementation protocol and as under certain circumstances such reactions are prone to errors which would cause false positive and negative results, care must be taken to avoid the errors. Errors commonly arise when sequences which generate or tolerate hairpins and internal loops, mismatch hybridization

, shifted mismatches and 3'-end hybridization are used. Application of appropriate sequence considerations, such as base composition and melting temperature (T_m) of hybrids have proven to be very effective in other experimental contexts where mis-hybridization was a critical consideration [8, 10]. In order to minimize use of error-prone sequences, genetic algorithms can be employed to optimize their design. In the protocol of this paper, we use a genetic algorithm to produce DNA sequences. The multiobjective fitness function employed in this algorithm is designed with respect to the above consideration [14, 20]. We believe the algorithm produces DNA sequences which have minimal potentials for errors and which can be used reliably for computations. Similarly, as ligation reactions are included in the implementation protocol, care must be taken to avoid mis-ligations. The use of an appropriate ligase under stringent conditions will prevent mis-ligations [2, 3, 16].

Under the above encoding, the machine is implemented as follows. First we create the DNA strands associated to the input alphabet, states and transition function of *M*. The sequences are designed using the genetic algorithm and taking care that the double stranded form of all have similar melting temperatures. Later, the start state molecule $HL(\bar{x}_0)$ is synthesized such that the chemical moiety biotin is linked to its 3′ terminus. $HL(\bar{x}_0)$ is immobilized onto streptavidin-coated paramagnetic beads. The strong attraction between the biotin on the DNA molecule and the streptavidin on the beads facilities this immobilization [18]. Consider $W=w_0w_1\ldots w_{n-1}$ as an input string; for each w_k $(k=0,\ldots,n-1)$, the following steps are repeated consecutively:

1. All the strands: 5'-$x_i+a_{w_k}$-3'$(0\le i\le v-1)$, corresponding to the following transition function elements: $d(q_i,w_k)=q_j$ $(0\le i, j\le v-1)$, are added to the beads in the presence of DNA ligase. Under suitable conditions for hybridization and ligation reactions, complementary strands are hybridized and adjacent strands are ligated, thus creating double-stranded DNA molecules. Some of the double stranded molecules are bound to the beads. The paramagnetic beads to which DNA is attached are removed from the suspension by use of a magnet, leaving unbound DNA strands in solution.
2. All the strands: 5'-$HR(\bar{x}_i)+\bar{a}_{w_k}+HL(\bar{x}_j)$-3' $(0\le i, j\le v-1)$, corresponding to the following transition function elements: $d(q_i,w_k)=q_j$ $(0\le i, j\le v-1)$, are added to the beads in the presence of ligase under suitable conditions. Appropriate strands again become hybridized and ligated and connected to the DNA on the beads. The beads are removed from the suspension as before, again leaving unbound DNA strands in solution.

In any *k*th iteration, before Step 1, a double-stranded DNA with a single-stranded overhang, which specifies the current state, is linked to the beads. During Step 1 and Step 2, strands whose sequences are dictated by w_k are added to the beads and some may become ligated to the existing bound strands to create longer strands. The newly created strands also have a single-stranded overhang. The overhang specifies the next state of transition function which, as has been shown, was reached as a consequence of having scanned w_k.

For the extraction process, fluorescently labeled DNA strands which represent the left half of the final states are added to the beads. After separation of the beads from the solution containing unbounded DNA strands, the beads are assessed for fluorescence, the presence of which symbolizes acceptable of the input string by the finite state automata. Thus, detection of

computational results is optical.

2. Computationally Simulated Molecular Algorithm

We developed a tool for simulating our genetic algorithm for sequence generation and our molecular algorithm discussed in this paper. This tool is executed for evaluating the automata shown in Figure 1. The processes of our simulation is explained by the following stages.

First, the following DNA codes are assigned to each state and input alphabet: x_0=5'-CAGCATAG-3', x_1=5'-ACACGTAG-3', x_2=5'-CAGAGAGT-3', a_0=5'-CATCTGAG-3', and a_1=5'-ACGTCATC-3', where x_0, x_1, and x_2, respectively, correspond to q_0, q_1, and q_2 states, and a_0 and a_1, respectively, correspond to $s_0=0$ and $s_1=1$ alphabaet. Here the defined length of DNA strands is assumed to be eight nucleotides. Now the transition function elements are coded as follows:

$d(q_i, s_k) = q_j$	5'-$x_i + a_{s_k}$-3'	3'-$HR(\bar{x}_i) + \bar{a}_{s_k} + HL(\bar{x}_j)$- 5'
$d(q_0, s_0) = q_0$	CAGCATAG CATCTGAG	TATC GTAGACTC GTCG
$d(q_0, s_1) = q_1$	CAGCATAG ACGTCATC	TATC TGCAGTAG TGTG
$d(q_1, s_0) = q_2$	ACACGTAG CATCTGAG	CATC GTAGACTC GTCT
$d(q_1, s_1) = q_1$	ACACGTAG ACGTCATC	CATC TGCAGTAG TGTG
$d(q_2, s_0) = q_0$	CAGAGAGT CATCTGAG	CTCA GTAGACTC GTCG
$d(q_2, s_1) = q_1$	CAGAGAGT ACGTCATC	CTCA TGCAGTAG TGTG

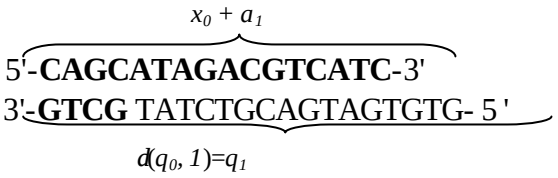
Then, the biotin-linked complement of the left half of the DNA strand corresponding to the start state q_0, $HL(\bar{x}_0)$, is immobilized onto streptavidin-coated paramagnetic beads. Therefore, initially GTCG is bound to the beads. Now, we describe the stages of our algorithm for the string $W=100$. In the first iteration w_0=1 is scanned, and we perform the following operations.

1. According to w_0, all the DNA strands: $x_i + a_1$ ($0 \le i \le 2$) are added to the beads. Under suitable conditions for hybridization and ligation, only $HL(\bar{x}_i)$ and $x_0 + a_1$ become hybridized and the remaining strands are discarded after removal of the beads. Therefore bound to the beads, we have:

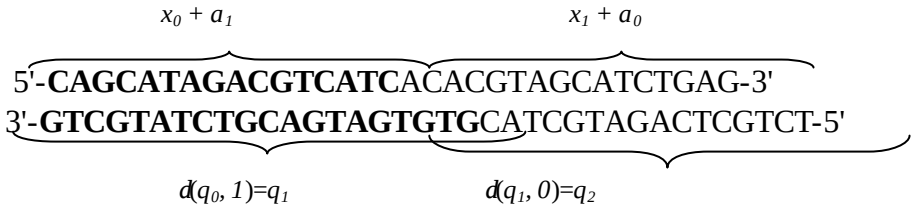
$$\overbrace{\text{5'-CAGCATAGACGTCATC-3'}}^{x_0 + a_1}$$
$$\text{3'-GTCG-5'}$$

2. All the strands: $HR(\bar{x}_i) + \bar{a}_1 + HL(\bar{x}_j)$ ($0 \le i, j \le 2$) are added to the beads. Again,

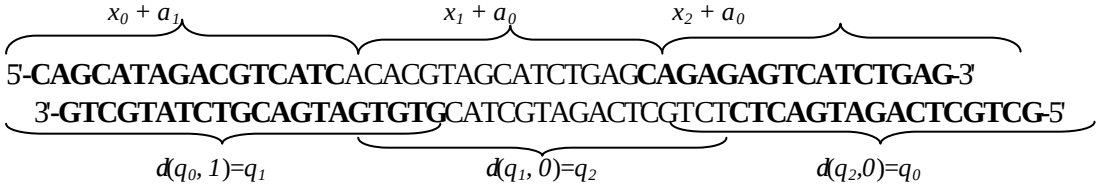
hybridization and ligation can occur. $HR(\bar{x}_0) + \bar{a}_1 + HL(\bar{x}_1)$ becomes hybridized to the DNA already on the beads. After removal from unbound strands, we have the following double strand molecule attached to the beads:



The second iteration is performed similarly to the first and the double stranded DNA molecule bound to the beads after accomplishing this iteration is:



The DNA on the beads after performing the third iteration is:



Now the left half of final state q_0, $HL(x_0)$, labeled with a flurescent dye, is added to the beads. This strand becomes hybridizes and ligated to the DNA perviously bound to the beads. The beads, after removal from suspension, are optically screened for the presence of fluorescence. Fluorescence is detected which means the input string is accepted by this automata.

2. Modified Implemented Finite State Machines

At the end of the algorithm defined in the previous section, the DNA bound to the beads which is checked for rejection or acceptance, is a double-stranded molecule whose length depends on the length of the input string. With long input string lengths, the increasing length of the DNA attached to the beads may impede the progress of required reactions. With minimal modifications, this problem can be circumvented. With regard to sequence design, the strands corresponding to the alphabets are re-designed such that they all end with *biotin*-CCC and the DNA strands corresponding to the states of the machine are also re-designed to begin with *biotin*-GGG. As a consequence, after hybridization and ligation of the strands in accordance with the algorithm, a *SmaI* restriction enzyme recognition site $\begin{pmatrix} 5'\text{-CCCGGG-}3' \\ 3'\text{-GGGCCC-}5' \end{pmatrix}$ is created at

the junction between an alphabet sequence and a state sequence. The double-stranded DNA molecules can be cleaved at this junction point by *SmaI*. The presence of biotin will not interfere with the hybridization, ligation, and restriction enzyme cleavage reactions. With regard to implementation, the algorithm of the pervious section needs to be modified by the additional of two steps. First, *SmaI* restriction enzyme must be added to the DNA linked beads recovered from step two, resulting in cutting of the DNA molecules at the alphabet-state junctions in the center of *SmaI* recognition site. Second, the beads should now be removed using a magnet and the released double-stranded DNA molecules with an overhang left behind, immobilized via their biotin groups onto new streptavidin-coated paramagnetic beads. In the first *SmaI* reaction, the enzyme will act on the following molecule bound to the beads:

$$5'\text{-}\ x_i + a_{w_k} + HL(x_j) + HR(x_j) + a_{w_{k+1}}\text{-}3'$$
$$3'\text{-}\ \bar{x}_i + \bar{a}_{w_k} + HL(\bar{x}_j)\text{-}5'$$

The product of the reaction released from the beads will be:

$$5'\text{-}HL(x_j) + HR(x_j) + a_{w_{k+1}}\text{-}3'$$
$$3'\text{-}HL(\bar{x}_i)\text{-}5'$$

These will again be bound to beads. Clearly, the length of strands attached to the beads becomes independent of the length of the input string.

5. Implementation of Nondeterminism

Nondeterminism is supposed to be well understood in the context of finite state machines, since the well-known subset construction [19] produces a deterministic equivalent of a given nondeterministic finite state machine. Nondeterministic finite state machines can also be implemented by DNA molecules. This can be done easily by our algorithm and only the number of strands used in the algorithm needs to be increased. Since the number of target states reachable from any state increases, the number of double-stranded DNA molecules on the beads corresponding to the transition function will also increase. Eventually, only one of these double-stranded molecules will specify the rejection or acceptance of the input string.

6. Conclusion

Two new models are presented for implementing finite state machines with DNA computing. In implementing the first model, the size of the molecules representing the finite state control depends on the length of the input string. In addition, the only limitation on sequences of input strands corresponding to the alphabet and the states of finite state machine are related to error minimalization considerations. The three operations used in this algorithm, ligation, hybridization and optical extraction, are all very simple. The use of biotin labelled DNA and streptavidin-coated paramagnetic beads allows each cycle to be performed within minutes manually. But a very attractive feature of the system is its compatibility with DNA chip technology thus readily lending itself to automation, which would allow hundreds of thousands

input sequences to be analyzed simultaneously and rapidly. By use of different colored fluorescent dyes, different questions can potentially be addressed simultaneously. In the second model, impediments caused by increasing lengths of the input string are eliminated. An enzymatic reaction is added to the operations of the first model, as a result of which the length of the DNA attached to the beads remains unchanged before and after each step of algorithm and, therefore, independent of the length of the input string. In this case, alphabet and state sequence design is limited in that the nucleotide sequences in the beginning and the end of strands should contribute to creation of a restriction site.The other molecular operations used in the second model are similar to the first model. Consequently, these two models are simple models to implement in the laboratory. The implementations of both models can be further made fault-tolerant and can be easily used for implementing nondeterministic models.

References

[1] L. Adleman, Molecular computation of solutions to combinatorial problems. *Science* **266** (1994), 1021-1029.

[2] D. O. Antson, A. Isaksson, U. Landegren, and M. Nillson, PCR generated padlock probes detect single nucleotide variation in genomic DNA. *Nucl. Acids Res.* **28** (2000), E58.

[3] F. Barany, Genetic disease detection and DNA amplification using cloned thermostable ligase. *Proc. Natl. Acad.* Sci. **88** (1991), 189--193.

[4] D. Beaver, Computing with DNA. *J. of Comput. Biol* **2** (1995), 1-13.

[5] 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.* **100** (2003), 2191-2196.

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

[7] R. S. Braich, N. Chelyapov, C. Johnson, P. W. K. Rothemund, and L. Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* **296** (2002), 499-502.

[8] R. Deaton, M. Garzon, R. C. Murphy, J. A. Rose, D. R. Franceschetti, and S. E. Stevens Jr, Reliability and efficiency of a DNA based computation. *Phys. Rev. Lett.* **80** (1998), 417-420.

[9] Y. Gao, M. Garzon, R. C. Murphy, J. A. Rose, R. Deaton, D. R. Franceschetti, and S.

E. Stevens Jr, DNA implementation of nondeterminism. In: H. Rubin, D. H. Wood (eds.), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **48**, American Mathematical Society, Providence, 1999, 137-148.

[10] P. Hardenbol, J. Baner, M. Jain, M. Nilsson, E.Namsaraev, G. Karlin-Neumann, H. Fakhrai-Rad, M. Ronaghi, T. Willis, U. Landegren, and R. Davis, Multiplexed genotyping with sequence tagged molecular inversion probes. *Nat. Biotechnol*. **21** (2003), 673-678.

[11] A. Hjelmfelt, E. Weinberger, and J. Ross, Chemical implementation of neural networks and Turing machines. *Proc. Natl. Acad. Sci.* **88** (1991), 10983-10987.

[12] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory*, languages and computation. Addison-Wesley, New York, 2000.

[13] Z. Ibrahim, Y. Tsuboi, O. Ono, and M. Khalid, Direct-proportional length-based DNA computing for shortest path problem. International Journal of *Computer Science & Applications* **1** (2004), 46-60.

[14] D. Kim, S. Shin, I Lee, and B. Zhang, NACST/Seq: a sequence design system with multiobjective optimization. *Lecture Notes in Computer Science* **2568**, Springer-Verlag, London, 2002, 242-251.

[15] R. J. Lipton, Speeding up computations via molecular biology. *Science* **268** (1995), 542-544.

[16] M. Nilsson, H. Malmgren, M. Samiotaki, M. Kwiatkowski, B.Chowdhary, and U. Landegren, Padlock probes: Circularizing oligonucleotides for localized dna detection. *Science* **265** (1994), 2085-2088.

[17] M. Ogihara and A. Ray, Simulating Boolean circuits on DNA computers. *Algorithmica* **25** (1999), 239-250.

[18] N. Pourmand, E. Elahi, R. W. Davis, and M. Ronaghi, Multiplex pyrosequencing. *Nucl. Acids Res*. **30** (2002), E31.

[19] P. Rothemund, A DNA restriction enzyme implementation of turing machines. In: E. B. Baum, R. J. Lipton (eds.), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **27**, American Mathematical Society, Providence, 1996, 75-119.

[20] F. Tanaka, M. Nakatsugawa, M. Yamamoto, T. Shiba, and A.Ohuchi, Developing support system for sequence design in DNA computing.*Lecture Notes in Computer Science* **2340**, Springer-Verlag, London, 2001, 129-137.

[21] M. Yamamoto, N. Matsuura, T. Shiba, Y. Kawazoe, and A.Ohuchi, Solutions of shortest path problems by concentration control. In: N.Jonoska, N. C. Seeman (eds.), *Lecture Notes in Computer Science* **2340**, Springer-Verlag, Berlin, 2002, 203-212.