TUHH
Technische Universität Hamburg-Harburg

Technical Report Series
Computer Engineering
Department

Israel M. Martínez-Pérez, Zoya Ignatova, and
Karl-Heinz Zimmermann

# An Autonomous DNA Model for Finite State Automata

May, 2006 Technical Report 06.1

# An Autonomous DNA Model for Finite State Automata

Israel M. Martínez-Pérez⋆[1], Zoya Ignatova⋆⋆[2], and Karl-Heinz Zimmermann[1]

[1] Institute for Computer Technology, Hamburg University of Technology, 21071 Hamburg, Germany
[2] Cellular Biochemistry, Max Planck Institute for Biochemistry, Martinsried, Germany

**Abstract.** In this paper we introduce an autonomous DNA model for finite state automata. This model called sticker automata model is based on the hybridization of single stranded DNA molecules (stickers) encoding transition rules and input data. The computation is carried out in an autonomous manner by one enzyme and will allow to determine whether a resulting double-stranded DNA molecule belongs to the automaton's language or not.

## 1 Introduction

One of the outstanding challenges in DNA computing is the design of autonomous, programmable DNA devices. A few models of this kind have been proposed like DNA self-assembly [12, 13], hairpin engines for NP-complete problems [8, 10], Whisplash PCR [6], 2-state 2-symbol finite state machine [3, 4], and 2-state 5-colour universal Turing machine [14]. These models differ by the degree of implementability, autonomy, programmability, and energy saving.

Recently, in a major breakthrough, Benenson *et al.* [5] designed an automaton capable of *in vitro* diagnostics and therapeutics of cancer. With such kind of automata, one could imagine to solve various decision making problems at the molecular level not only in life science, but also in agriculture and biotechnology. This would amount to a definite killer application of DNA computing where classical computers could not compete with.

In this paper we propose an alternative autonomous DNA model for general $n$-state $m$-symbol automata. This model is called sticker automata model as the transition rules and input data are encoded by single stranded DNA molecules (stickers). The sticker automata model will allow (theoretically) to implement all kind of applications for finite automata at the molecular level.

## 2 The Shapiro Model

The DNA model suggested in Benenson *et al.* [5] implements a two-state two-symbol automaton. A finite state automaton is a model of computation consisting of a set of states, an input alphabet, an initial state, a set of transition rules (each of which mapping the current state and the actual input symbol to the next state), and a set of final states. A finite state automaton can decide whether an input string is accepted or not. To this end, the finite state automaton performs a computation beginning with the initial state reading the first symbol from the input string. The computation consists of a series of transitions. In each transition, the next input symbol is read from the input string and the current state is changed according to the transition

⋆ Supported by CONACYT and DAAD.
⋆⋆ Supported by DFG Heisenberg Foundation.

rules to establish a new state. If no transition rule applies, the process may be suspended without completing the computation. The computation terminates when the automaton has read the last symbol from the input string. The automaton will accept the input string if it terminates in an accepting final state. The language of the automaton is the set of all accepted input strings over the input alphabet (Fig. 1).

The Shapiro model is composed of three parts: input data, software, and hardware. The hardware consists of a class IIS restriction endonuclease (*Fok*I). The software comprises a set of short double-stranded DNA molecules implementing the transition rules of the automaton. Input data and initial state are encoded by double-stranded DNA molecules. The model also contains two output detection molecules (one per state).

The computation proceeds in a controlled manner over the input molecule processing one symbol at a time either until a terminator is reached or no transition rule can be applied. To this end, the double-stranded region of a transition rule contains the enzyme recognition site and its sticky end encodes the current state and symbol to be processed. The next state is defined by a series of nucleotides between the recognition site and the sticky end of the transition rule molecule. When a transition rule molecule binds to the input data molecule, the enzyme cuts the double-stranded DNA at a constant distance from the recognition site leaving a sticky end which in turn encodes the next state and symbol to be processed. This process continues until no further reaction can take place. The final sticky end molecule may anneal to an output detector molecule to form an output reporter molecule. This will reveal that the input string is accepted.

The Shapiro model operates in a complete autonomous fashion, and does not require ATP, which makes it an energy saving device. On the oher hand, the Shapiro automaton is a two-state two-symbol machine and so has a very limited complexity (number of symbols times number of states). Any increase in the complexity is bounded from above by the number of different non-palindromic sticky ends. The discovery or engineering of new class IIS enzymes with longer spacers and/or longer sticky ends might allow the implementation of automata with higher complexity.

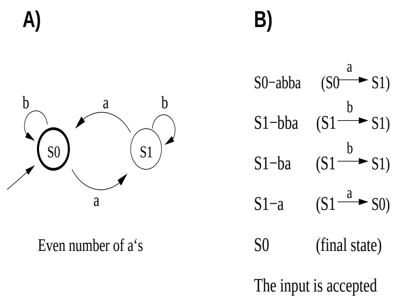


**Fig. 1.** A two-state two-symbol finite automaton with input alphabet {a,b}. $S_0$ is the initial and final state. Labeled arrows represent transition rules. The automaton's language is the set of all strings over $\{a, b\}$ which contain an even number of $a$'s. The figure is adapted from Benenson *et al.* [3].

## 3 The Sticker Automata Model

As Shapiro's model, our DNA model of a finite state automaton is composed of three parts: input data, software, and hardware. Despite of this similarity to Shapiro's model, there are significant differences. Firstly, in our DNA model, input data and software are encoded by single stranded DNA (stickers). Secondly, the software does not contain any recognition site for restriction enzymes. Thirdly, the hardware is composed of one enzyme (either Mung Bean or S1) different from Shapiro's hardware. Fourthly, symbols and states of the automaton are separately encoded.

## 3.1 Representation of Information

An input string is encoded by a single stranded DNA molecule which has the following form: initiator (5'-terminus), alternating sequence of symbols and spacers beginning and ending with a spacer, and terminator (3'-terminus). Initiator, terminator, spacers and all symbols of the automaton's alphabet are encoded by single stranded DNA sequences. Let $S_1, \ldots, S_n$ denote the states of the automaton. A spacer encodes the states by a single stranded DNA sequence in which $n$ equally spaced locations correspond to the states in a predefined order (Fig. 2a).

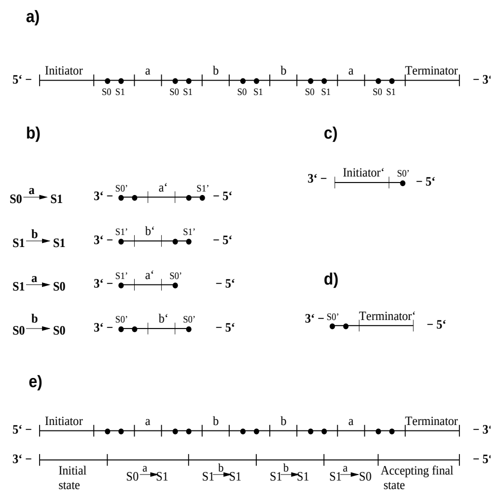
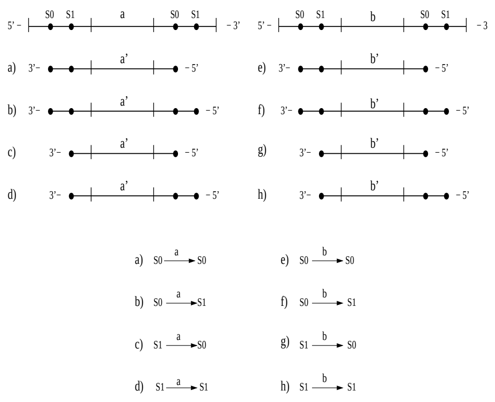


**Fig. 2.** Schematic encoding of the automaton given in Fig. 1. (a) Encoding of the input string $abba$. (b) Encoding of the four transition rules. (c) Encoding of initial state. (d) Encoding of accepting final state. (e) Schematic representation of the double stranded DNA molecule encoding the accepted input string $abba$.

The transition rules form the heart of the sticker automata model (Fig. 2b). A transition rule $S_{\text{current}} \xrightarrow{\text{symbol}} S_{\text{next}}$ is given by an oligonucleotide encoded by the Watson-Crick complementarity of the 3' $S_{\text{current}}$ part of the spacer, the symbol, and the 5' $S_{\text{next}}$ part of the spacer (Fig. 3).

An initial state oligonucleotide is encoded by the Watson-Crick complementarity of the initiator followed by the 5' $S$ part of the space corresponding to the initial state $S$ (Fig. 2c). A final state oligo is encoded by the Watson-Crick complementarity of the 3' $S$ part of the spacer associated with the final state $S$ and the terminator (Fig. 2d).



5' – S0 S1 a S0 S1 – 3'
5' – S0 S1 b S0 S1 – 3
a) 3'– a' – 5'
b) 3'– a' – 5'
c) 3'– a' – 5'
d) 3'– a' – 5'
e) 3'– b' – 5'
f) 3'– b' – 5'
g) 3'– b' – 5'
h) 3'– b' – 5'
a) $S0 \xrightarrow{a} S0$
b) $S0 \xrightarrow{a} S1$
c) $S1 \xrightarrow{a} S0$
d) $S1 \xrightarrow{a} S1$
e) $S0 \xrightarrow{b} S0$
f) $S0 \xrightarrow{b} S1$
g) $S1 \xrightarrow{b} S0$
h) $S1 \xrightarrow{b} S1$

**Fig. 3.** Encoding of the eight possible transition rules of a two-state two-symbol automaton.

## 3.2 Operation of the Automaton

The operation of the molecular automaton consists of three steps: Data pre-processing, computation, and output verification.

The data pre-processing consists of the annealing and linking of all single stranded DNA molecules which are encodings of input data, transition rules, initial states, and accepting final states. After data pre-processing, the accepted input strings will correspond to ***complete*** double stranded DNA molecules as shown in Fig. 2e, while the non-accepted input strings will correspond to ***partial*** double stranded DNA (Fig. 4).

The computation is carried out by either Mung Bean Nuclease or S1 Nuclease. While Mung Bean degrades single stranded DNA with extremely low exonuclease activity, S1 nuclease degrades single-stranded nucleic acids. Mung Bean Nuclease is preferable to S1 Nuclease for most applications because it has lower intrinsic activity on duplex DNA [7]. Therefore, both enzymes can degrade the single stranded region of a non-accepted input string. As a consecuence, complete double stranded DNA molecules corresponding to accepted input strings will remains intact after digestion.

We could use gel electrophoresis to detect a DNA molecule corresponding to an accepted input string according to its length. However, this would require knowledge of the lengths of the molecules in advance. On the other hand, the DNA molecule of an accepted input string has both, an initiator and terminator sequence. So PCR may be used to detect molecules corresponding to accepted input strings.

## 4 Discussion

At this moment, we are implementing in laboratory 2-state and 4-state automata using the sticker automata model. As soon as we have the first results, it will be included in the paper. If we suceed, the generality of the model would allow to implement all kinds of applications for finite automata at the molecular level (at least in theory). The computational power would be greater than Shapiro's automata, because this model can encode as many states and symbols as needed. The complexity is bounded from above by the number of different non-palindromic encodings of the variables and by the physical limitations of DNA.

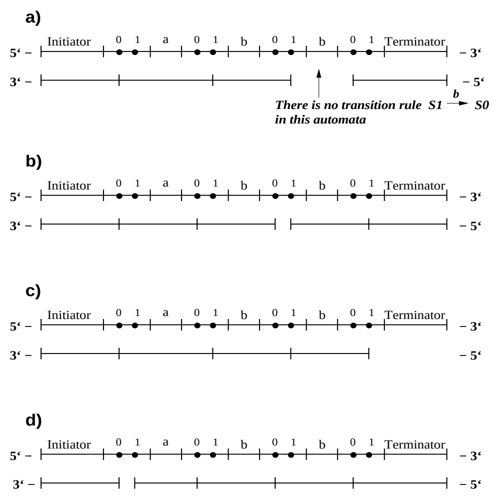


**Fig. 4.** Annealing cases for non-accepted input strings. There are three annealing cases leading to the rejection of an input string (illustrated by the automaton in Fig. 1): (a) non-existing transition rule, (b) illegal transition rule: S0-b-S0 followed by S1-b-S1, (c) non-accepting final state, and (d) illegal initial state.

Moreover, it seems possible to keep the same conditions of autonomy and energy saving like in the Shapiro model. The computation is carried out in an autonomous manner by the nucleases, and we do not need ATP in this part of the process. The only part where ATP is required is in the data pre-processing, equivalent to the assembly of the machine components in Shapiro's model. On the other hand, the high probability of mismatched annealing between transition rules and input data could significantly decrease the efficiency of the computation in the sticker automata model.

## References

1. R. Adar, Y. Benenson, G. Linshiz, A. Rosner, N. Tishby and E. Shapiro, Stochastic computing with biomolecular automata, *Proc. Natl Acad. Sci. USA*, 101, 2004, 9960-9965.
2. R. Bar-Ziv, T. Tlusty, A. Libchaber, Protein-DNA computation by stochastic assembly cascade, *Proc. Natl Acad. Sci. USA*, 99, 2002, 11589-11592.
3. 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.
4. Y. Benenson, T. Paz-Elizur, R. Adar, Z. Livneh and E. Shapiro, DNA molecule provides a computing machine with both data and fuel, *Proc. Natl Acad. Sci. USA*, 100, 2003, 2191-2196.
5. Y. Benenson, B. Gil, U. Ben-Dor, R. Adar and E. Shapiro, An Autonomous molecular computer for logical control of gene expression, *Nature*, Adv. online publ., 2004.
6. M. Hagiya, K. Sakamoto, M. Arita, D. Kiga and S. Yokoyama, Towards parallel evaluation and learning of boolean u-formulas with molecules, *Proc. 3rd DIMACS Meeting DNA Based Computers*, 1997.
7. W.D. Kroeker, Mung bean nuclease I. Terminally directed hydrolisis of native DNA, *Biochemistry*, 15, 1976, 4463-4467.
8. I.M. Mart´ınez-P´erez, G. Zhong, Z. Ignatova, and K.-H. Zimmermann, Solving the Hamiltonian path problem via DNA hairpin formation, *Int. J. Bioinformatics Research Applications*, 1, 2005, 389-398.
9. D. Luo, The road from biology to materials, *Materialstoday*, 2003, 38-43.
10. K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori and M. Hagiya, Molecular computation by DNA hairpin formation, *Science*, 288, 2000, 1223-1226.
11. K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, S. Ikeda, H. Sugiyama and M. Hagiya, State transitions by molecules, *Biosystems*, 52, 1999, 81-91.

12. E. Winfree, F. Liu, L.A. Wenzler and N.C. Seeman, Design and self-assembly of two-dimensional DNA crystals, *Nature*, 394, 1998, 539-544.
13. E. Winfree, X. Yang, N.C.. Seeman, Universal Computation via self-assembly of DNA: Some theory and experiments, *Proc. 2nd DIMACS Meeting DNA Based Computers*, 1996.
14. P. Yin, A. Turberfield, S. Sahu and J.H. Reif, Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion, *Science*, Adv. online publ., 2004.