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 Stochastic Finite State Automata

May, 2006

Technical Report 06.2

An Autonomous DNA Model for Stochastic 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

1 Introduction

Stochastic computing has several important applications in science and engineering such as artificial neural networks [10], stochastic analysis of biological information [6], simulation of probabilistic processes and their simulation [8], probabilistic reasoning [8], population behaviour with random walkers [16], stochastic dynamic programming [13].

2 Stochastic Finite Automata

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 mappingthe current state and the actual input symbol to the next state), and a set of final states. The transitions rules may be given by a function or a relation, mapping or relating the current state and the actual input symbol to the next state. In the first case, the automaton is called deterministic and in the second, non-deterministic. 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 rules to establish a new state. 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). For each non-deterministic finite state automaton there exists a deterministic finite state automaton such that both accept the same language. So non-deterministic finite state automata are not more powerful than determinstic ones.

A finite state automaton is called stochastic if the transition rules are defined by transition probabilities and initial and final states are defined by probability distributions. A stochastic finite state automaton A consists of an input alphabet Σ, a finite state set S, an initial probability distribution q_0 on state set S, i.e., $\sum_s q_0(s) = 1$, a conditional probability distribution $P(\cdot \mid a, s)$ on state set S for each pair $(a, s) \in \Sigma \times S$, i.e., $\sum_{s'} P(s' \mid a, s) = 1$ for all $(a, s) \in \Sigma \times S$, and a final probability distribution q_f on state set S, i.e., $\sum_s q_f(s) = 1$ (Fig. 2).

A stochastic finite automaton A can decide whether an input string is accepted or not. For this, A perform a computation by reading the first symbol of the input string starting in the initial state s with probability

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

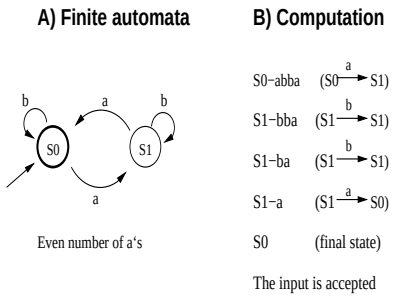

Fig. 1. A two-state two-symbol finite automaton with input alphabet {a,b}. S_0 is the initial and final state. Labeled arrows represent transition rules. The automaton's language is the set of all strings over $\{a, b\}$ which contain an even number of a's. The figure is adapted from Benenson *et al.*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

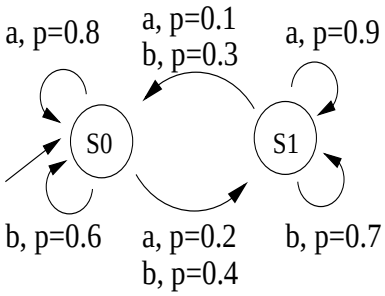

Fig. 2. A two-state two-symbol automaton with input alphabet {a,b}. S_0 is the initial and final state, i.e., $q_0(S_0) = q_f(S_0) = 1$ and $q_0(S_1) = q_f(S_1) = 0$. Labeled arrows represent transition probabilities. The figure is adapted from Adar *et al.* [1].

$q_0(s)$. The computation consists of a series of iterations. In each iteration, the next input symbol a is read. If A is in state s with probability q, then with probability $P(s' \mid a, s) \cdot q$ it enters the state s'. The computation terminates when A has read the last symbol from the input string.

Formally, the behaviour of A can be described by extending the transition probabilities so that for the empty word ϵ,

$$P(s' \mid \epsilon, s) = \begin{cases} 1 \text{ if } s = s', \\ 0 \text{ if } s \neq s', \end{cases}$$

and for each nonempty word $\boldsymbol{a} = a_1 \dots a_{n+1}$ over Σ and states $s, s' \in S$,

$$P(s' \mid a_1 \dots a_n a_{n+1}, s) = \sum_{s''} P(s' \mid a_{n+1}, s'') P(s'' \mid a_1 \dots a_n, s).$$

That is, in order to reach the state s' from the state s by reading $\boldsymbol{a}$, all intermediate states s'' need to be considered that can be reached from s when the substring $a_1 \dots a_n$ is read.

It follows that $q_f(s')P(s' \mid \boldsymbol{a}, s)q_0(s)$ is the probability that A reaches the final state s' when it reads input string $\boldsymbol{a}$ starting in the initial state s. For instance, in view of the parser in Fig. 2, the probability of the input string ab is

$$\begin{aligned} & q_f(S_0)P(S_0 \mid ab, S_0)q_0(S_0) = \\ & = q_f(S_0) \left[P(S_0 \mid b, S_0)P(S_0 \mid a, S_0) + P(S_0 \mid b, S_1)P(S_1 \mid a, S_0)\right] q_0(S_0) \\ & = 1.0 \cdot [0.6 \cdot 0.8 + 0.3 \cdot 0.2] \cdot 1.0 = 0.54. \end{aligned}$$

The term $\sum_{s,s'} q_f(s')P(s' \mid \boldsymbol{a}, s)q_0(s)$ is the probability of $\boldsymbol{a}$ in A, i.e., the probability that A enters the final states when it reads $\boldsymbol{a}$ starting from the initial states.

A stochastic finite automaton A becomes a stochastic finite parser by taking into account a threshold value $\lambda \in [0, 1]$. A stochastic finite parser A will accept an input string if the probability of being in the final states exceeds the threshold value. More formally, the language $L_\lambda(A)$ of A with given threshold value λ is given by all input strings whose probability in A exceeds λ, i.e.,

$$L_\lambda(A) = \{\boldsymbol{a} \mid \sum_{s,s'} q_f(s')P(s' \mid \boldsymbol{a}, s)q_0(s) > \lambda\}.$$

For instance, the language of the parser in Fig. 2 consists of all input strings $\boldsymbol{a}$ with probability $P(S_0 \mid \boldsymbol{a}, S_0) > \lambda$ as S_0 is the only initial and final state (Fig. 3).

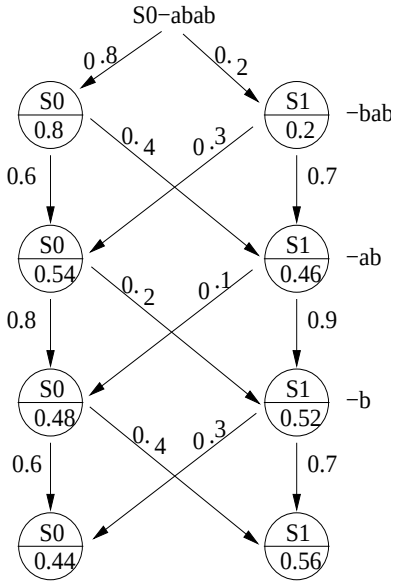

Fig. 3. A computation of the stochastic finite automaton A in Fig. 2. The string $abab$ has the probability 0.44 in A. The figure is adapted from Adar *et al.* [1].

3 The Sticker Stochastic Automata Model

Our DNA model of stochastic finite state automata will extend our previous DNA model of finite state automata [11]. It is also composed of three parts: input data, software, and hardware. Our model differs from Shapiro's model in several respects. 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. 4a).

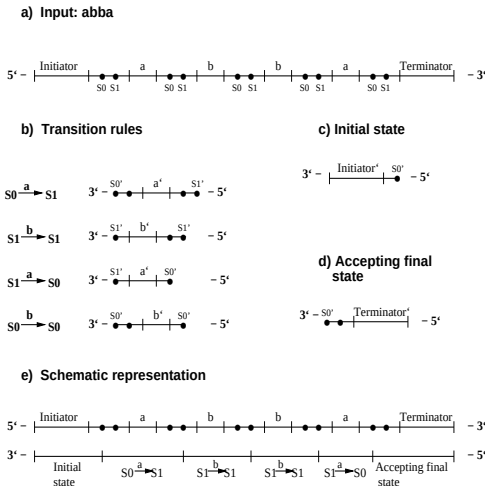

Fig. 4. Schematic encoding of the automaton given in Fig. 1. (a) Encoding of the input string $abba$. (b) Encoding of the four transition rules. (c) Encoding of initial state. (d) Encoding of accepting final state. (e) Schematic representation of the double stranded DNA molecule encoding the accepted input string $abba$.

The transition rules form the heart of the sticker automata model (Fig. 4b). A transition rule $S_{\text{current}} \overset{\text{symbol}}{\longrightarrow} S_{\text{next}}$ is given by an oligonucleotide encoded by the Watson-Crick complementarity of the 3' S_{current} part of the spacer, the symbol, and the 5' S_{next} part of the spacer (Fig. 5).

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. 4c). 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. 4d).

3.2 Operation of the Automaton

Our molecular automaton operates in 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. The intended probabilities of the transitions are realized by the relative molar concentrations of the molecules encoding the transitions. Similarly, the intended probabilities of the initial and final states are implemented by the relative molar concentrations of the corresponding oligonucleotides encoding the initial and final states, respectively.

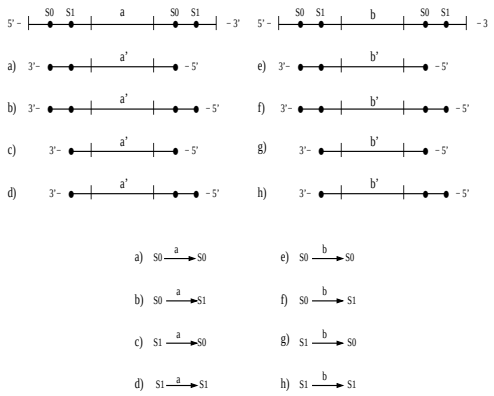

Fig. 5. Encoding of the eight possible transition rules of a two-state two-symbol automaton.

After pre-processing, we can distinguish between two kinds of input strings. Firstly, there are input strings which are completely processed by the automaton and correspond to ***complete*** double stranded DNA molecules as shown in Fig. 4e. Secondly, there are input strings which are not completely processed by the automaton and correspond to ***partial*** double stranded DNA (Fig. 6).

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 [9]. Therefore, both enzymes can degrade the single stranded region of a non-accepted input string. As a consecuence, only the complete double stranded DNA molecules corresponding to completely processed input strings will remains intact after digestion.

Finally, the relative concentration of the DNA molecules corresponding to completely processed input strings can be measured. In particular, for given threshold λ, we need to measure the relative molar concentration of the DNA molecules corresponding to the accepting and non-accepting input strings. This requires a relation between the relative molar concentrations and the threshold value.

4 Discussion

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*, Advanced online publication, 2004.
6. R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchinson, *Biological Sequence Analysis: Probabilistic Models of Proteins and Amino Acids*, Cambridge Univ. Press, Cambridge, 1998.

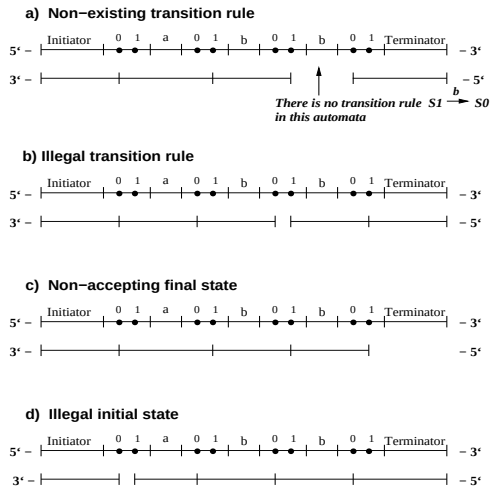

Fig. 6. Annealing cases for non-accepted input strings. There are three annealing cases leading to the rejection of an input string (illustrated by the automaton in Fig. 1): (a) non-existing transition rule, (b) illegal transition rule: S0-b-S0 followed by S1-b-S1, (c) non-accepting final state, and (d) illegal initial state.

7. 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.
8. H. Hansson and B. Jonsson, A logic for reasoning about time and reliability, *Formal Aspects of Computing*, 6, 1994, 512-535.
9. W.D. Kroeker, Mung bean nuclease I. Terminally directed hydrolisis of native DNA, *Biochemistry*, 15, 1976, 4463-4467.
10. S.Y. Kung, Digital Neural Networks, Prentice Hall, 1997.
11. 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.
12. D. Luo, The road from biology to materials, *Materialstoday*, 2003, 38-43.
13. S.M. Ross, *Introduction to Stochastic Dynamics Programming*, Academic Press, San Diego, 1983.
14. 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.
15. 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.
16. R.V. Sole, E. Bonabeau, J. Delgado, P.Fernandez, and J. Marin, Small-world behaviour in a system of mobile elements, *Europhysics Letters*, 53, 2001, 693-699.
17. 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.
18. 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.
19. 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*, Advanced online publication, 2004.

Technical Reports in this Series:

Technical Report 93.6 (November 1993)
O. Knüppel: A Multiple Precision Arithmetic for PROFIL

Technical Report 93.7 (December 1993)
S. M. Rump: Verification Methods for Dense and Sparse Systems of Equations

Technical Report 94.1 (February 1994)
C. Jansson, O. Knüppel: Numerical Results for a Self-Validating Global Optimization Method

Technical Report 94.2 (June 1994)
C. Jansson: On Self-Validating Methods for Optimization Problems

Technical Report 94.3 (December 1994)
C. Jansson: Some Properties of Linear Interval Systems and Applications

Technical Report 95.1 (March 1995)
S. M. Rump: Improved Iteration Schemes for Validation Algorithms for Dense and Sparse Nonlinear Systems

Technical Report 95.2 (March 1995)
S. M. Rump: Expansion and Estimation of the Range of Nonlinear Functions

Technical Report 95.3 (May 1995)
S. M. Rump: Bounds for the Componentwise Distance to the Nearest Singular Matrix

Technical Report 95.4 (October 1995)
O. Knüppel: A PROFIL/BIAS Implementation of a Global Minimazation Algorithm

Technical Report 95.5 (October 1995)
S. M. Rump: Perron-Frobenius like Theorems for not Sign-Restricted Matrices

Technical Report 96.1 (July 1996)
S. M. Rump: Almost sharp bounds on the componentwise distance to the nearest singular matrix

Technical Report 96.2 (August 1996)
S. M. Rump: Theorems of Perron-Frobenius type for matrices without sign restrictions

Technical Report 96.3 (November 1996)
K.-H. Zimmermann: Integral Hecke Modules, Integral Generalized Reed-Muller Codes, and Linear Codes

Technical Report 96.4 (December 1996)
C. Jansson, J. Rohn: An Algorithm for Checking Regularity of Interval Matrices

Technical Report 97.1 (February 1997)
K.-H. Zimmermann, G. Ehlers, W. Brandt, F. A. M. Bouchard, J. Diedrichsen, T. Möller: Das ELLA 2000 Mikrocontroller-Projekt

Technical Report 97.2 (June 1997)
K.-H. Zimmermann: A Class of Double Coset Codes

Technical Report 97.3 (November 1997)
C. Jansson: An NP-hardness result for nonlinear systems

Technical Report 98.1 (March 1998)
C. Jansson: Construction of convex lower and concave upper bound functions

Technical Report 98.2 (July 1998)
G. Ehlers, K.-H. Zimmermann: Einführung in den Hardware-Entwurf mit ELLA

Technical Report 98.3 (October 1998)
S.M. Rump: Fast and Parallel Interval Arithmetic

Technical Report 98.4 (October 1998)
S.M. Rump: INTLAB – Interval Laboratory

Technical Report 99.1 (March 1999)
O. Knüppel: PROFIL/BIAS V 2.0

Technical Report 99.2 (March 1999)
J.-P. M. Zemke: b4m – A free interval arithmetic toolbox based on BIAS

Technical Report 99.3 (October 1999)
N. Suresh Babu, Karl-Heinz Zimmermann: A Short Introduction to Quaternary Codes

Technical Report 99.4 (October 1999)
C. Jansson: Quasiconvex Relaxations Based on Interval Arithmetic

Technical Report 00.1 (October 2000)
Karl-Heinz Zimmermann: DNA Algorithms for Binary Linear Codes

Technical Report 01.1 (January 2001)
Karl-Heinz Zimmermann: The Patch Selection Problem and Discrete Stochastic Dynamic Programming

Technical Report 01.2 (March 2001)
Tangha Tina Lai, Karl-Heinz Zimmermann: A Software Platform for the Sticker Model

Technical Report 01.3 (June 2001)
Prashant Batra: Omitted Values and Real Robust Schur Stability

Technical Report 02.1 (June 2002)
C. Jansson: Rigorous Lower and Upper Bounds in Linear Programming

Technical Report 03.1 (January 2003)
C. Jansson: A rigorous lower bound for the optimal value of convex optimization problems

Technical Report 04.1 (January 2004)
T. Ogita, S.M. Rump, and S. Oishi: Accurate Sum and Dot Product

Technical Report 04.2 (June 2004)
Prashant Batra: On Small Circles Containing Zeros and Ones of Analytic Functions

Technical Report 05.1 (February 2005)
Israel M. Martínez-Pérez, Zoya Ignatova, Zhang Gong, Ming-Woei Lau, Karl-Heinz Zimmermann: Self-Assembly of DNA Molecules

Technical Report 05.2 (April 2005)
Israel M. Martínez-Pérez, Zoya Ignatova, Zhang Gong, Ming-Woei Lau and Karl-Heinz Zimmermann: A Programmable Computing Machine Operating Autonomously at the Molecular Scale

Technical Report 06.1 (May 2006)
Israel M. Martínez-Pérez, Zoya Ignatova, and Karl-Heinz Zimmermann: An Autonomous DNA Model for Finite State Automata

Technical Report 06.2 (May 2006)
Israel M. Martínez-Pérez, Zoya Ignatova, and Karl-Heinz Zimmermann: An Autonomous DNA Model for Stochastic Finite State Automata