Sebastian Sakowski, Tadeusz Krasiński, Joanna Sarnik, Janusz Blasiak, Jacek Waldmajer and Tomasz Poplawski*

A detailed experimental study of a DNA computer with two endonucleases

DOI 10.1515/znc-2016-0137
Received March 1, 2017; revised March 1, 2017; accepted March 26, 2017

Abstract: Great advances in biotechnology have allowed the construction of a computer from DNA. One of the proposed solutions is a biomolecular finite automaton, a simple two-state DNA computer without memory, which was presented by Ehud Shapiro's group at the Weizmann Institute of Science. The main problem with this computer, in which biomolecules carry out logical operations, is its complexity – increasing the number of states of biomolecular automata. In this study, we constructed (in laboratory conditions) a six-state DNA computer that uses two endonucleases (e.g. *Acu*I and *Bbv*I) and a ligase. We have presented a detailed experimental verification of its feasibility. We described the effect of the number of states, the length of input data, and the nondeterminism on the computing process. We also tested different automata (with three, four, and six states) running on various accepted input words of different lengths such as *ab*, *aab*, *aaab*, *ababa*, and of an unaccepted word *ba*. Moreover, this article presents the reaction optimization and the methods of eliminating certain biochemical problems occurring in the implementation of a biomolecular DNA automaton based on two endonucleases.

Keywords: biomolecular computers; DNA computing; finite automata.

***Corresponding author: Tomasz Poplawski,** Department of Molecular Genetics, University of Lodz, Pomorska 141/143, 90-236 Lodz, Poland, E-mail: tomasz.poplawski@biol.uni.lodz.pl
Sebastian Sakowski and Tadeusz Krasiński: Faculty of Mathematics and Computer Science, University of Lodz, Banacha 22, 90-238 Lodz, Poland
Joanna Sarnik and Janusz Blasiak: Department of Molecular Genetics, University of Lodz, Pomorska 141/143, 90-236 Lodz, Poland
Jacek Waldmajer: Department of Philosophy, University of Opole, Katowicka 89, 45-061 Opole, Poland

1 Introduction

The tremendous advances in genetics and molecular biology have enabled the use of DNA as a nanomaterial for computation. Most researchers in DNA computing, the new computational paradigm that uses biomolecules for computations [1], have applied well-known biological operations to solve particular problems in computer science [2–4]; in addition, they have also designed nanomachines built on DNA [5, 6]. DNA computing has found implementation in medicine, e.g. in the diagnosis and therapy of cancer [7] as well as in in vitro DNA-based logic circuits and gates to diagnose the levels of miRNA [8]. Another approach focuses on the potential application of biocomputing machines to biological processes [9, 10] and to monitor enzymatic regulation [11]. A technique called DNA manipulation has been used in the biomolecular implementation of digital circuits [12] as well as neural networks [13]. Research into DNA computations have focused on the biomolecular implementation of logical operations [14, 15], molecular inference [16, 17], and molecular programming [18]. Over a period of a few decades of research into DNA computing, various theoretical models have been proposed, for example, the splicing system [1, 19], the self-assembly model [20], computing with membranes [21], and the sticker-based model [22]. The self-assembly of DNA molecules has enabled the folding of DNA molecules into shapes such as squares and five-pointed stars [23]; this DNA origami has allowed the coupling of molecular emitters to photonic crystal cavities [24]. Current research in DNA computing shows sophisticated computer systems, e.g. biomolecular calculators that operate by selecting the results from a library [25] or using biomolecular memory as an alternative storage media [26].

Although many studies have focused on the biomolecular implementation of known theoretical models of computation [27–29], only a few of them have presented detailed reports on laboratory applications. This asserts the necessity of creating interdisciplinary research groups composed of researchers from different areas of science, such as computer scientists and biologists, who can solve some complex, biochemical-related problems. For this

reason, the laboratory test of a biomolecular nondeterministic finite automaton (a simple two-state two-symbol automaton) [30] is considered a major event in DNA computing.

A deterministic finite automaton (DFA) represents the simplest model of a computer [31] that enables us to solve plain problems (operations of addition and multiplication of integers are beyond the capabilities of such machines). Each finite automaton (FA) consists of a tape with cells containing an input word created from symbols of a certain finite alphabet, and a control unit reading the symbols of the input word one after another and changing its state according to the transition rules. Each transition rule is of the form $(s_0, a) \rightarrow s_1$ when an automaton is in the state s_0, after reading the symbol a, it transits to the state s_1. The automaton accepts the input word if starting from the initial state, after reading the whole word, its control unit transits to one of the distinguished final states. Usually, FA are represented by graphs [32]. Figure 1B presents an example (in the form of graph structure) of a two-state automaton, which accepts words with an odd number of a's. A variant of FA is the nondeterministic finite automata (NFA). The difference between DFA and NFA is that every state of a DFA always has exactly one exiting transition rule for each symbol in the alphabet, whereas in NFA, a state may have zero, one, or many exiting transition rules for each symbol in the alphabet. NFA accepts an input word when one of the possible ways leads to an accepting state and reads the whole input word. An example of an NFA is given in Figure 1C (accepting words with at least one symbol a).

In 2001, a group at the Weizmann Institute of Science implemented a two-state two-symbol NFA [30] (see all possible transition molecules in Figure 1A) that used only one restriction enzyme (RE), *Fok*I and *ligase*. In later studies, researchers showed that molecular computers can act without *ligase* in some limited manner [33, 34]. These first implementations were based on the use of the following biomolecular elements: *Fok*I RE, ligase enzyme, and double-stranded DNA fragments (an input molecule, a set of transition molecules, and a set of detection molecules). An input molecule (Figure 2A) is a double-stranded DNA molecule that represents an input word (for example, the input word $x = aba$) consisting of the symbols a and b. A single-stranded overhang (a sticky end) in the input word represents not only a symbol but also a state, i.e. a pair <*state*, *symbol*>. Another element of the biomolecular DNA automaton is a set of transition molecules (software). Each transition molecule contains a specific sequence recognizable by the *Fok*I, a single-stranded overhang (a sticky end), and the additional base pairs (Figure 2B for one transition $(s_0, a) \rightarrow s_1$). An integral part of such automatons are the detection molecules (detectors, Figure 2C), which in laboratory experiments recognize the final state of computation. These molecules consist of a single-stranded overhang (sticky end) and an additional double-stranded fragment of DNA.

If we want to make a computation, first we choose (fix) an NFA, i.e. we chose some transitions from all possible transition molecules (Figure 1A). Next, we put all elements together: *Fok*I, many copies of the chosen transition molecules, detection molecules, and the input molecules. In each cycle of the computation process (Figure 3),

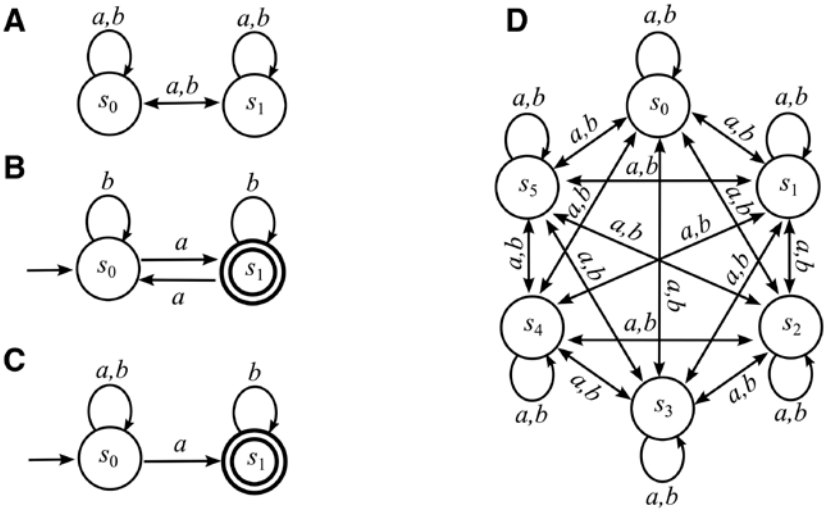

Figure 1: The FA. (A) All possible transitions of a two-state two-symbol NFA. (B) An example of a two-state two-symbol DFA. (C) An example of a two-state two-symbol NFA. (D) All possible transitions of a six-state two-symbol NFA.

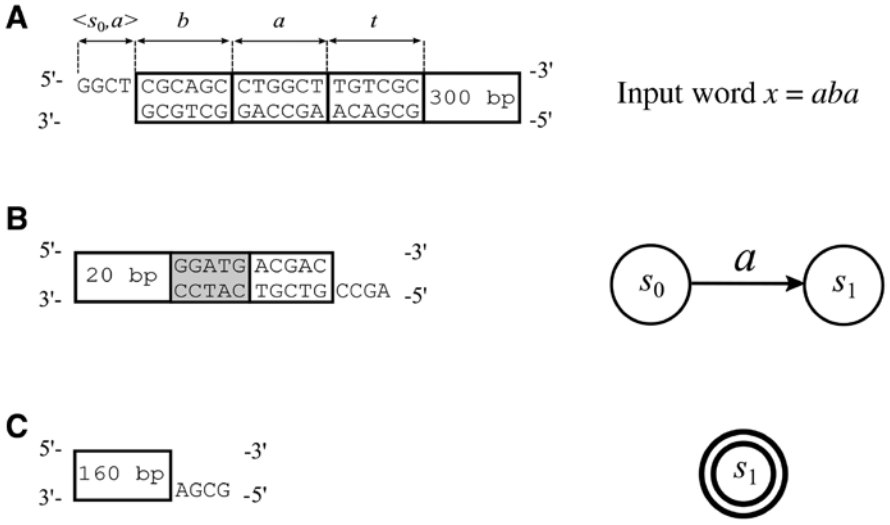

Figure 2: The elements of a biomolecular FA and corresponding elements of the FA in the form of a graph: (A) an example of an input molecule representing the input word *aba*. (B) An example of a transition rule. (C) An example of detection molecule for state s_1.

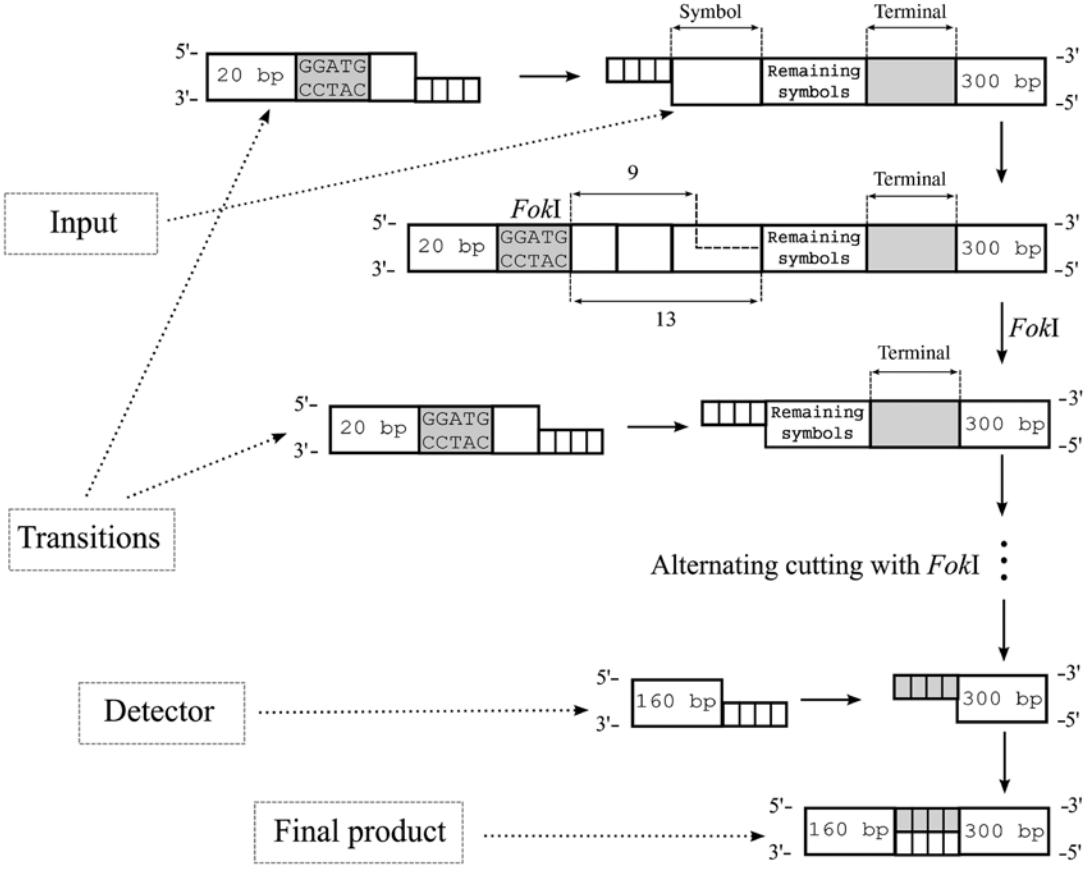

Figure 3: Schematic diagram of a computation using one endonuclease *Fok*I presented in 2001 by Benenson et al. [30].

a transition molecule is combined with a sticky end of the input molecule and *Fok*I can cut inside the next symbol and reveal a new sticky end. Such operations precisely reflect the action of an NFA.

To increase the complexity of the DNA computer presented by Ehud Shapiro's group at the Weizmann Institute of Science [30], we designed a theoretical model of a six-state, two-symbol NFA (see Figure 1D). DNA molecules

representing all 72 transitions of the six-state, two-symbol FA are presented in the Appendix of our previous article [35]. The sticky ends cleaved by the REs *Bbv*I and *Acu*I are four and two nucleotides in length, respectively. A general computation scheme consists of alternating the cutting and ligating processes of the DNA molecules using *Bbv*I and *Acu*I (Figure 4). In this article, we built and tested the more powerful six-state automaton. It is composed of our earlier theoretical ideas [35, 36] and it is, to our knowledge, the first working example of a six-state automaton built with DNA and REs. We tested all aspects that were important for DNA computer works: the effect of the length of

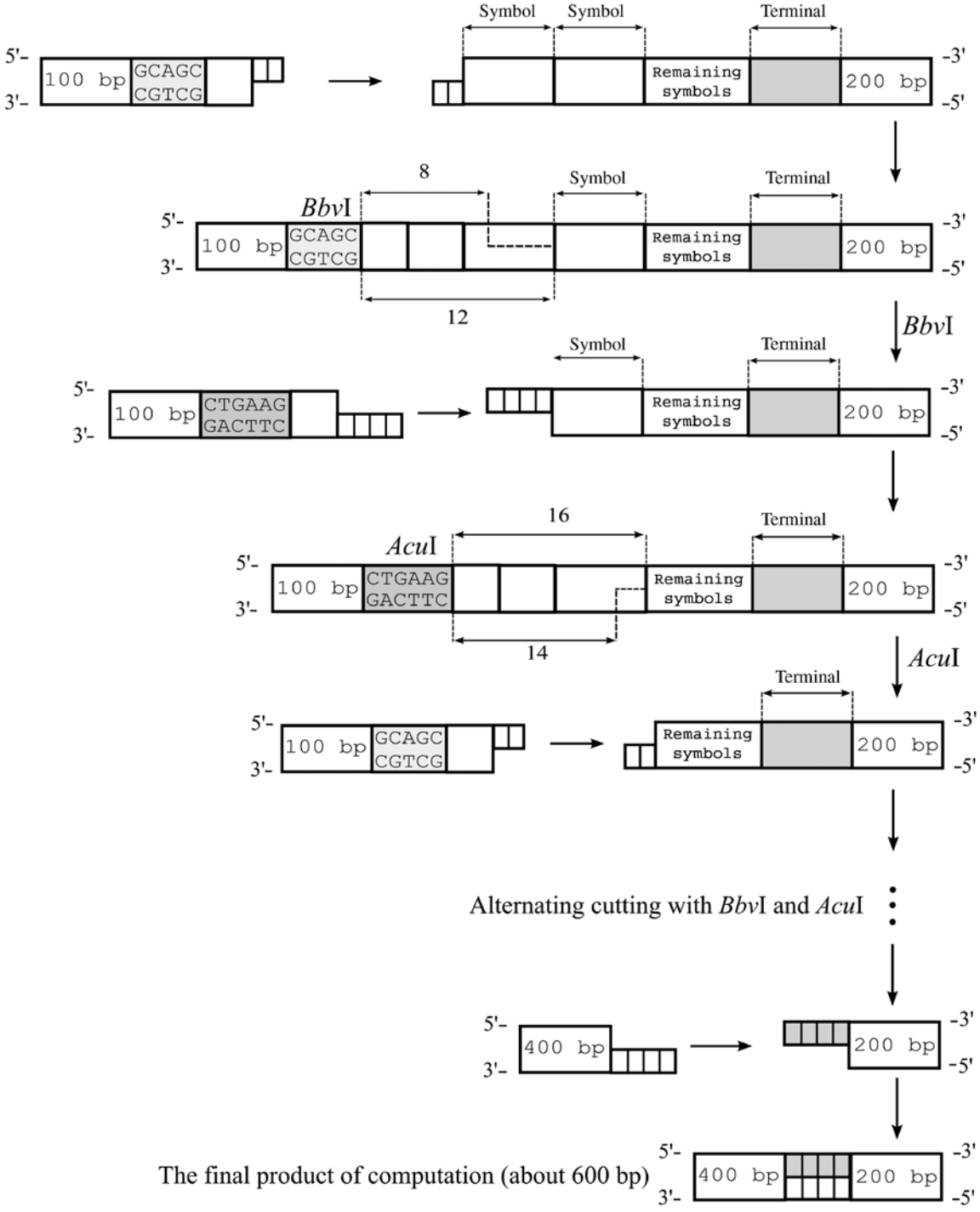

Figure 4: Schematic diagram of a computation using many endonucleases, e.g. *Bbv*I and *Acu*I.

input words on the result of the experiment as well as the effect of nondeterminism of finite state automata on the results. We also performed experimental tests of six-state automaton, reaction optimization, and eliminated certain biochemical problems.

2 Materials and methods

2.1 Synthetic DNA

Synthetic oligonucleotides (lyophilized, 200 nmol) were from Genomed (Warsaw, Poland). The oligonucleotides for software (transition molecules) were ST56_1 (5′-AATTCTGAAGGTAGAAGGTATTAGTTGCTCCTC-3′), ST56_2 (5′-GGCCGAGGAGCAACTAATACCTTCTACCT-TCAG-3′), ST50_1 (5′-AATTCTGAAGCATCTTCATCATTAGTTGCTCCTC-3′), ST50_2 (5′-GGCCGAGGAGCAACTAATGATGAAGATGCTTCAG-3′), ST42_1(5′-AATTGCAGCTCTGAATTAGTTGCTCCTC-3′), ST42_2 (5′-GGCCGAGGAGCAACTAATTCAGAGCTGC-3′), ST67_1 (5′-AATTGCAGCTCTCTCCGATTAGTTGCTCCTC-3′), ST67_2 (5′-GGCCGAGGAGCAACTAATCGGAGAGAGCTGC-3′), ST66_1 (5′-AATTCTGAAGTCTCTCGCTGTATTAGTTGTCATCGC-3′), ST66_1 (5′-GGCCGCGATGACAACTAATACAGCGAGAGACTTCAG-3′), ST2_1 (5′-AATTGCAGCTCTGTCGTATTAGTTGTCATCGC-3′), ST2_2 (5′-GGCCGCGATGACAACTAATACGACAGAGCTGC-3′), T32_1 (5′-CGGCAGCGGTAGTAACCATTATTCATCGC-3′), T32_2 (5′-GCGATGAATAATGGTTACTACCGCTGC-3′), T65_1 (5′-CGCTGAAGGGTAGGCTGAACCATTATTCATCGC-3′), T65_2 (5′-GCGATGAATAATGGTTCAGCCTACCCTTCAG-3′), T8_1 (5′-CGCTGAAGGGTAGAATGCGATTAGTTGCTCCTC-3′), T8_2 (5′-GAGGAGCAACTAATCGCATTCTACCCTTCAG-3′), T66P_1 (5′-CGCTGAAGGTAGAAGCTGTATTAGTTGTCA-TCGC-3′),T66P_2 (5′-GCGATGACAACTAATACAGCTTCTACCTTCAG-3′), T67P_1 (5′-CGGCAGCGTAGAACGATACTTTAGATTGACTTCAG-3′), T67P_2(5′-CTGAAGTCAATCTAAAGTATCGTTCTACGCTGC-3′).

The oligonucleotides for the construction of the input molecules (input word *ab, ba, aab, aaab, ababa*) and detection molecule were SAB_1 (5′-CGCCAATTATCAGCCGATACTTTAGATTGCCTTCAG-3′), SAB_2 (5′-CTGAAGGCAATCTAAAGTATCGGCTGATAATTGG-3′), SBA_1 (5′-CGCCAATTCGACTAATATACTTTAGATTGCCTTCAG-3′), SBA_2 (5′-CTGAAGGCAATCTAAAGTATATTAGTCGAATTGG-3′), SAAB_1 (5′-CGCCAATTATCAGCCGACTACGATACTTTAGATTGCCTTCAG-3′), SAAB_2 (5′-CTGAAGGCAATCTAAAGTATCGTAGTCGGCTGATAATTGG-3′), SAAAB_1 (5′-CGCCAATTATCAGCCGACTACGACTACGATACTTTAGATTGCCTTCAG-3′), SAAAB_2 (5′-CTGAAGGCAATCTAAAGTATCGTAGTCGTAGTCGGCTGATAATTGG-3′), SABABA_1 (5′-AATTCCAATTCGACTAATCAGCCGACTAATCAGCCGATTAGTTGCTCCTC-3′), SABABA_2 (5′-GGCCGAGGAGCAACTAATCGGCTGATTAGTCGGCTGATTAGTCGAATTGG-3′), TERM_5_1 (5′-CGTTATACTTTAGATTGCCTTCAG-3′), TERM_5_2 (5′-CTGAAGGCAATCTAAAGTATAA-3′), TERM_4_1 (5′-CGTGATACTTTAGATTGCCTTCAG-3′), TERM_4_2 (5′-CTGAAGGCAATCTAAAGTATCA-3′), STERM_1_1 (5′-AATTATTGTATTAGTTGTCATCGC-3′), STERM_1_2 (5′-GGCCGCGATGACAACTAATACAAT-3′).

The transition molecules, input molecule, and detection molecule were prepared by annealing pairs of oligonucleotides (sense and antisense): ST56_1 and ST56_2, ST50_1 and ST50_2, ST42_1 and ST42_2, ST66_1 and ST66_1, ST67_1 and ST67_2, ST2_1 and ST2_2, T32_1 and T32_2, T65_1 and T65_2, T8_1 and T8_2, T66P_1 and T66P_2, T67_1 and T67_2, T67P_1 and T67P_2, SAB_1 and SAB_2, SBA_1 and SBA_2, SAAB_1 and SAAB_2, SAAAB_1 and SAAAB_2, SABABA_1 and SABABA, TERM_5_1 and TERM_5_2, TERM_4_1 and TERM_4_2, STERM_1_1 and STERM_1_2.

2.2 Enzymes and plasmid vectors

*Bbv*I and *Acu*I, which were used as the hardware and to prepare DNA molecules representing the components of the automaton, *Bsm*AI, *Cla*I, *Bsr*DI, *Btgz*I, *Not*I and *Acu*I, and T4 DNA ligase, were from New England Biolabs (Ipswich, MA, USA). T4 polynucleotide kinase (PNK), pJET 1.2 and LITMUS 38i plasmids were from Fermentas (Grand Island, NY, USA).

2.3 Chemicals

Plasmid Miniprep Kit and Gel Extraction Kit were from Axygen (Union City, CA, USA). Perfect 100 bp DNA Ladder was from Eurx (Gdansk, Poland). It contained 13 bands with fragments of the following sizes: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, and 2500 bp. For easy reference, bands at 500 bp and 1000 bp were considered brighter than other bands in the ladder. Thermo Scientific GeneRuler 1 kb DNA Ladder is composed of 14 chromatography-purified individual DNA fragments (in base pairs): 10,000, 8000, 6000, 5000, 4000, 3500, 3000, 2500, 2000, 1500, 1000, 750, 500, and 250 bp. It contained three reference bands of (6000, 3000, and 1000 bp) for easy orientation. Sigma-Aldrich pUC18 DNA *Hae*III Ladder contained 11 bands with fragments of the following sizes: 587, 458, 434, 298, 267, 257, 174, 102, 80, 18, and 11 bp. All other chemicals and bacterial media were from Sigma-Aldrich (Saint Louis, MO, USA).

2.4 Construction of DNA libraries containing DNA molecules for use in the computational reactions

The general scheme of preparing automaton's components was different from that presented by Ehud Shapiro's group at the Weizmann Institute of Science [30]. We built a "DNA library", a collection of DNA molecules representing computer elements that are stored and propagated in a population of *Escherichia coli* through the process of molecular cloning. We used two different experimental schemes labeled as experimental Scheme 1 and Scheme 2. Scheme 1 was used to test the three-state automaton with the use of the words: *ab*, *ba*, *aab*, and *aaab*. Scheme 2 was used to test four-state and six-state automata. The first part of the cloning procedure was the same in both experimental schemes. Single-stranded oligonucleotides were labeled according to the represented components of the automaton: the input words, the transition molecules, and detection molecules. These oligonucleotides were 5′ phosphorylated by the T4 PNK. We used 100 pmol of oligonucleotides, which were phosphorylated with 10 U PNK in 20 µl of the PNK buffer and 1 mM ATP, for 60 min at 37°C, precipitated with ethanol and suspended in TE buffer (10 mM Tris–HCl, 1 mM EDTA, pH 8.0). The molecules of the automaton's components were prepared by mixing the appropriate pairs of oligonucleotides. Thereafter, complementary oligonucleotides (sense and antisense) were incubated for 10 min at 95°C, and then slowly (1°C/min) cooled to room temperature. Then, double-stranded DNA molecules were cloned into the pJET 1.2 plasmid vector (Scheme 1) or LITMUS 38i (Scheme 2). The obtained clones were sequenced at the Institute of Biochemistry and Biophysics (Warsaw, Poland) and underwent restriction analysis. Typical restriction analysis of obtained clones are presented in the Supplementary Material. Before the computational reaction, DNA fragments were obtained by the cleavage of plasmid DNA with REs, agarose gel electrophoresis, and subsequent purification with the QIAquick DNA Purification Kit (Scheme 1). In experimental Scheme 2, we utilized polymerase chain reaction (PCR) to obtain DNA molecules. The proper sticky ends were generated with the appropriate REs followed by purification through agarose gel electrophoresis.

2.5 Polymerase chain reaction

We used PCR to obtain DNA molecules used in our computer with Perpetual OptiTaq PCR Master Mix (Eurx, Gdansk, Poland). PCR primers used in this study were from Genomed (Warsaw, Poland) and they were Primer_2 (5′-CGTGGCTAGCGGGAAG-3′), Primer_3 (5′-ACCATGATTACGCCAAGCTA-3′), Primer_4 (5′-AGGAGAGCGCACGAGGGA-3′), Primer_5 (5′-CTCACTCATTAGGCACCC-3′), and Primer_6 (5′-TGCTGCAAGGCGATTAAGTT-3′).

The PCR mixture (25 µl) consisted of 1.25 U Perpetual OptiTaq DNA Polymerase, 1× reaction buffer (1.5 mM $MgCl_2$), 0.2 mM of each dNTP, and 0.5 µM of upstream and downstream primer. PCR conditions were as follows: initial denaturation step at 95°C for 3 min, 30 cycles at 95°C for 30 s and 30 s at 60°C annealing temperature, and at 72°C for 30 s. The final extension step was performed at 72°C for 5 min. The PCR was carried out using an MJ Research, INC thermal cycler, model PTC-100 (Waltham, MA, USA). After the PCR, samples were subjected to restriction digestion with proper RE. The digested products were resolved on a 2% agarose gel and stained with 0.5 µg/ml ethidium bromide.

Transition molecules were prepared with Primer_2 and Primer_3, and the length of their final version (after digestion with RE and gel purification) was around 110 bp. The term molecule was prepared with Primer_2 and Primer_4 and its final length was 404 bp. The word molecule was prepared with Primer_5 and Primer_6 and its final length was 230 bp.

2.6 Computation reaction

In the reaction, we verified autonomous and programmable cleavage of DNA molecules by two endonucleases in one test tube. This reaction was run for 2 h in NEB4 buffer at 37°C. The reaction tube contained a set of DNA fragments representing the input and transitions molecules, 1u *Bbv*I and *Acu*I as well as 10 U T4 DNA ligase. The control sample was similar to test samples except REs and ligase. The product of the reaction was purified with phenol, chloroform, and izoamyl alcohol (25:24:1), precipitated with ethanol, and separated by 2% agarose gel electrophoresis. The reactions started with the ligation of a transition molecule with the input word. After the cyclic reactions of digestion, the ligation of the final DNA fragment (the rest of the input molecule) with the terminal molecule followed, producing the output-reporting molecule – the length of the DNA fragment was the sum of the rest input and the terminal molecules. Detection of the DNA fragment with such length by gel electrophoresis followed by ethidium bromide staining indicates the acceptance of the input word by the automaton.

3 Results

In this section, we present the results of laboratory implementation of our automata. These experiments focused on the key elements, essential to the operation of automata with two REs: *Bbv*I and *Acu*I.

3.1 The reaction optimization

Our DNA computer did not work without ligase and the ligation step was crucial for efficient computation reaction. Thus, we had to optimize it to compromise the cutting and joining processes. The optimum temperature of the most commonly used T4 DNA ligase is around 37°C. Therefore, the best choice for blunt-ended DNA fragments is 37°C. However, sticky ends are often too short to form a stable duplex at these conditions; in that case, ligation at lower temperature (25°C) is preferable. We also used relatively short DNA fragments because joining contiguous DNA fragments (thousands of bp) is very challenging. We found the optimal condition for the biomolecular computer: the initial ligation step performed at 25°C (optimal temperature for T4 DNA ligase action) for 15 min, followed by incubation at 37°C (temperature for *Bbv*I, *Acu*I action) for 2 h, and the final ligation step at 25°C for 15 min.

3.2 The length of the input words has no effect on the DNA computer

We tested the operation of automaton A_1, presented in Figure 5A, by running it on various accepted input words of different lengths: *ab*, *aab*, *aaab*, and an unaccepted *ba*. The experiments presented in Figure 6A and B indicate that the length of the input words has no effect on the results of computation. Automaton A_1 worked as expected. In all the samples, we observed intermediate molecules (Figure 6A and B, lines 2 and 4). Intermediate molecules did not arise in the control samples that lacked enzymes (Figure 6A and B, lines 1 and 3). We also obtained the output-reporting molecule for the state s_5 (length 846 bp) in samples with the input words: *ab*, *aab*, and *aaab* input molecules (Figure 6A, line 2; Figure 6B, lines 2 and 4). The unaccepted input word *ba* was not computed, and we did not observe any output-reporting molecule for the state s_5 (length 846 bp; Figure 6A, line 4).

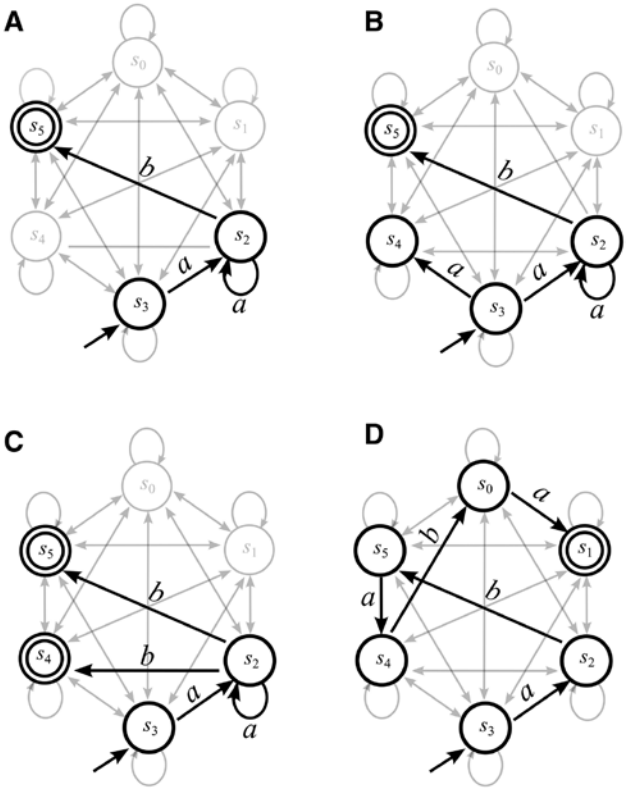

Figure 5: The finite automata testing in the laboratory: (A) FA A_1; (B) NFA A_2; (C) NFA A_3; (D) FA A_4.

3.3 The nondeterminism of the DNA computer

In these experiments, we had to slightly modify our computer in comparison to the previous one, as we planned to obtain two final states for the automaton in one tube. We changed the length of the molecules to distinguish both states. The output-reporting molecule for state s_5 was 714 bp in length, whereas for state s_4, it was 614 bp in length. We tested two NFAs, A_2 and A_3, presented in Figure 5B and C both on the same input word *aab*. In the automaton A_2, half the number of molecules representing the input word *aab* "chose" the wrong way through the transition $(s_3, a) \rightarrow s_4$. Nevertheless, the result for automaton A_2 (Figure 6C, lane 2) was almost the same as that of automaton A_3 (Figure 6D, lane 2) on the same input word *aab*. In automaton A_3, the nondeterminism appears in the final transitions. The results (Figure 6D, lane 2) confirmed the assumption that the input molecules "chose" both last transitions, $(s_2, b) \rightarrow s_5$ and $(s_2, b) \rightarrow s_4$, equally.

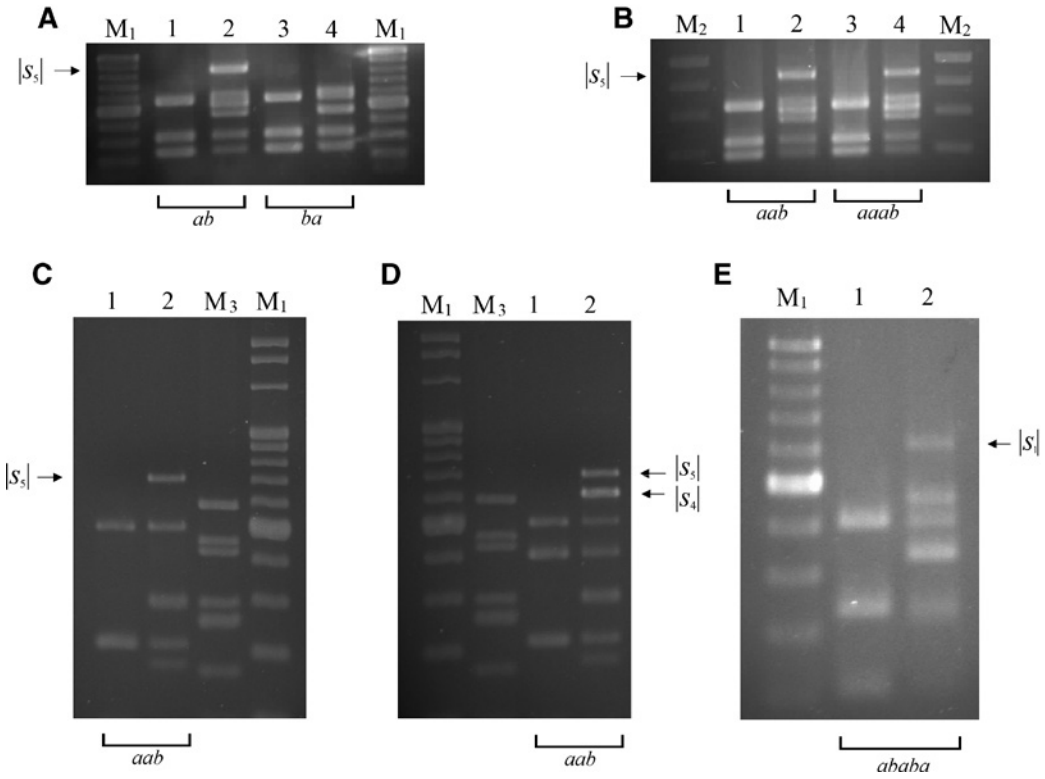

Figure 6: (A) Experimental testing of automaton A_1 on the accepted word *ab* and nonaccepted word *ba*. (B) Testing automaton A_1 on accepted inputs words: *aab*, *aaab*. (C) Experimental testing of the influence of nondeterminism of the automaton A_2 on results (on accepted input word *aab*). (D) Experimental testing of the influence of nondeterminism of automaton A_3 on results (on accepted input word *aab*). (E) Experimental testing of automaton A_4 on results (on accepted input word *ababa*). Abbreviations: M_1 – the size of the marker (100 base pair); M_2 – the size of the marker (1 kb); M_3 – the size of the marker (pUC *Hae*III); $|s_1|$ – the length of an output reporter (613 bp); $|s_4|$ – the length of an output reporter (614 bp); $|s_5|$ – the length of an output reporter (856 bp and 714 bp); 1,3 – the result of computation without endonucleases and without ligase; 2,4 – the result of computation with the use of endonucleases and ligase.

3.4 An experimental testing of the six-state automaton

We tested the operation of the six-state automaton A_4 presented in Figure 5D by running it on the accepted input word: *ababa*. The entire procedure is presented in Figure 7. The output-reporting molecule for state s_1 had a length of 613 bp. We validated the mechanism of operation by the results of two experiments: with REs (*Bbv*I, *Acu*I) and ligase; without REs (*Bbv*I, *Acu*I) and ligase. The results of the experiments (see Figure 6E) showed that it is possible to construct a six-state automaton using two REs (*Bbv*I and *Acu*I) and that this automaton could act autonomously in one test tube. Thus, we proved experimentally that by using two enzymes, e.g. *Bbv*I and *Acu*I, it is possible to expand the number of states (to six states) in biomolecular NFA, as presented by Ehud Shapiro's group at the Weizmann Institute of Science [30].

4 Discussion

In this article on our study, we have presented detailed laboratory tests of the biomolecular NFA with two endonucleases: *Bbv*I and *Acu*I. The experiments conducted have proved that it is possible to construct (in a wet lab) biomolecular models of computation, e.g. biomolecular NFA using REs and ligase. Additionally, this article shows that it is possible in a real laboratory to increase the number of states (to six states) in the biomolecular NFA presented by Ehud Shapiro's group at the Weizmann Institute of Science [30]. These experiments create an opportunity for the implementation of more complex NFA and other theoretical models of computers, e.g. pushdown automata [29].

The complexity (number of states) of biomolecular automata [30] is bounded from above by the number of available sticky ends. REs such as *Fok*I (with 4-nt sticky ends) enable the construction of automata with, at most,

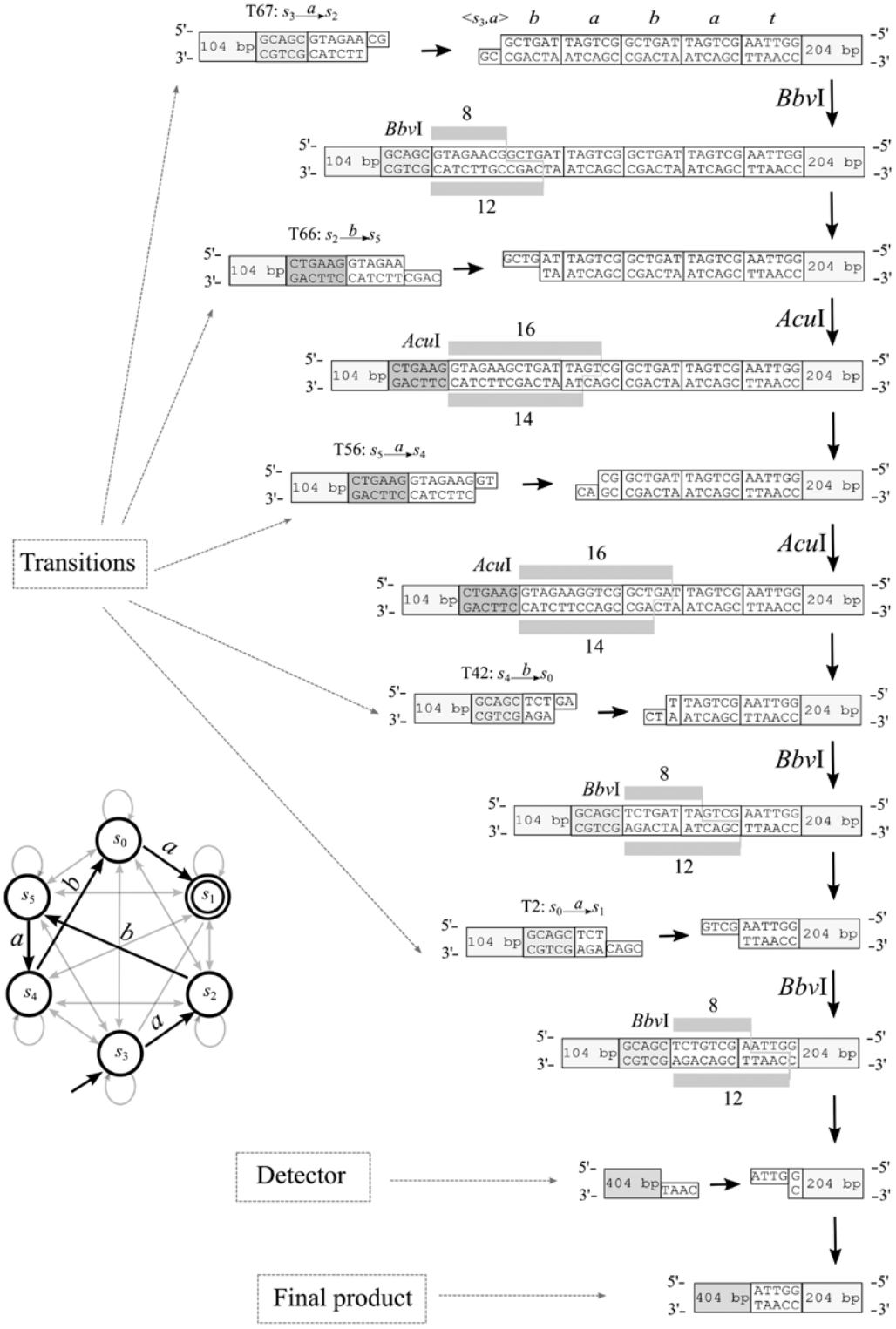

Figure 7: An example of a computation for six-state FA (the input word *ababa*) tested in the laboratory.

three states. Among extensions based on the use of one RE, one can distinguish the extension proposed by Unold et al. [37], as well as by Soreni et al. [38], whereas extensions based on the use of more than one RE were first proposed by us [35]. We also have considered and presented certain arithmetic estimates concerning the

possibilities of theoretical extension of a biomolecular DNA automaton using two or more REs [36]. An extended model based on many REs creates an opportunity to implement, in the laboratory, more complex NFA [39] and even other theoretical models of computers: pushdown automata [28] and Turing machines [27].

From the biochemical point of view, a computation reaction is the sequential joining and cutting of DNA molecules. In these biochemical reactions, we have distinguished some problems that could reduce the efficiency of a computation process. The real problem in this approach is the use of two different REs together (e.g. in the same test tube); therefore, we must fit them. Both the used REs are part of about 170 endonucleases available in FastDigest™ technology, of which digestion buffers support 100% activity of all endonucleases. Second, the efficiency of a reaction is being limited because of the problems related to the occurrence of many copies of DNA molecules in a single test tube. Analysis has revealed that some transition molecules may ligate with one another, i.e. two copies of the same transition molecule may ligate with one another. Such a situation can take place for the following two-nucleotide sticky ends (cleaved by the endonuclease *Acu*I) of transition molecules: AT, TA, GC, and CG because they are complementary to each other and ligate in reaction conditions. Further analyses have shown that, additionally, some different transition molecules ligate with one another when transition molecules were used simultaneously with the following two-nucleotide sticky ends: AA-TT, AC-GT, AG-CT, CA-TG, CC-GG, and GA-TC. The same situation applies for the sticky ends that are four nucleotides in length (cleaved by the endonuclease *Bbv*I), e.g. ATAT, TATA, GCGC, and CGCG. It is possible to eliminate such a situation by using different techniques to prevent self-ligation of DNA [40], e.g. standard dephosphorylation of the 5′ ends by alkaline phosphatase. The problem with two copies of the same transition molecule ligating with one another could be eliminated by expanding the genetic alphabet [41, 42] – using unnatural base pairs, e.g. isocytosine (dC), isoguanine (dG), 5-(2,4-diaminopyrimidine) (dj), and xanthosine (dX). In such a situation, sticky ends that are two nucleotides in length could be coded as AdC, dCA, dGC, CdG. This approach has the additional interesting property that the alphabet for codes of symbols could be expanded from four nucleotides (the alphabet is $\Sigma = \{A, T, G, C\}$) to, e.g. eight nucleotides (the alphabet is $\Sigma = \{A, T, G, C, dC, dG, dj, dX\}$). Thus, using unnatural base pairs, it will be possible to increase the complexity, i.e. the number of states and symbols of a biomolecular computer.

We should take into account not just the adding of more and more REs but also the unwanted cross-talk, the total noise of the system, even the limitations on finding universal buffers for all different enzymes, because any of these may affect the feasibility of the DNA computer. In our experiments, we tested the crucial elements of the biomolecular automaton's actions. First, we checked the autonomous actions of two REs (*Bbv*I and *Acu*I) in a test tube. Next, after successful preliminary tests, we checked the effect of the length of input words and nondeterminism of NFA on the results. Additionally, we tested various automata, e.g. a six-state biomolecular automaton. From these experiments, we obtained a main conclusion: "it is possible to construct more complicated finite automata using more than one RE". Moreover, the length of the input data and nondeterminism do not influence the detection of results. In addition, we have noticed that the efficiency of the biochemical process would decrease if we used sticky ends in the construction of biomolecular automatons, which ligate with one another. Therefore, we propose not to use sticky ends complementary with one another. Additionally, we suggest the possibility of using synthetic unnatural nucleotides in biomolecular implementation models of computation based on DNA. Such an approach to the design biomolecular computers allows us to eliminate biochemical problems occurring at the implementation of biomolecular computers.

We believe that using multiple and autonomous applications of REs in one test tube is the next step toward a complete understanding of their potential, and toward applying the computation with biomolecules in medicine, e.g. in the diagnosis and therapy of cancer. It should be noted that the DNA automata have potential to direct interaction with the biochemical world and may become the indicators of a disease within the living cell. The future of DNA automaton applications is currently connected with medical diagnosis, to the process of attempting to determine or to identify diseases such as cancer or other genetic disorders. The discovery of miRNA and the identification of their role in regulatory pathways controlling gene expression have opened new possibilities of using DNA automata as a cure [7]. Biomolecular computers may, in the future, communicate with biological systems and they have the potential to become excellent diagnostic tools.

Acknowledgements: This project is supported by the National Science Centre of Poland (NCN). Grant number: DEC-2011/01/B/ NZ2/03022.

References

1. Paun G, Rozenberg G, Salomaa A. DNA computing. New computing paradigms. Berlin, Heidelberg, New York: Springer, 1998.
2. Adleman L. Molecular computation of solutions to combinatorial problems. Science 1994;226:1021–4.
3. Faulhammer D, Cukras A, Lipton R, Landweber L. Molecular computation: RNA solutions to chess problems. Proc Natl Acad Sci USA 1999;97:1385–9.
4. Stojanovic M, Stefanovic D. Deoxyribozime-based molecular automaton. Nat Biotechnol 2003;21:1069–74.
5. Reif J, Sahu S. Autonomous programmable DNA nanorobotic devices using DNAzymes. Theor Comput Sci 2009;410:1428–39.
6. Sekiguchi H, Komiya K, Kiga D, Yamamura M. A realization of DNA molecular machine that walks autonomously by using a restriction enzyme. Lect Notes Comput Sci 2008;4848:54–65.
7. Benenson Y, Gil B, Ben-Dor U, Adar R, Shapiro E. An autonomous molecular computer for logical control of gene expression. Nature 2004;429:423–9.
8. Seelig G, Soloveichik D, Zhang D, Winfree E. Enzyme-free nucleic acid logic circuits. Science 2006;314:1585–8.
9. Muhammad MR, Pavan Kumar C, Selvakumar R. Generative and recognising devices for biological processes. Arab J Sci Eng 2016;41:2893–902.
10. Selvakumar R, Muhammad MR, Devi GP. An embedded automaton to monitor the glycolysis process in pancreatic β-cells. Acta Biotheor 2015;63:23–31.
11. Ali RM, Gurusamy PD, Ramachandran S. Computational regulatory model for detoxification of ammonia from urea cycle in liver. Turk J Biol 2014;38:679–83.
12. Qian L, Winfree E. Scaling up digital circuit computation with DNA strand displacement cascades. Science 2011;6034:1196–201.
13. Qian L, Winfree E, Bruck J. Neural network computation with DNA strand displacement cascades. Nature 2011;475:368–72.
14. Wąsiewicz P, Malinowski A, Nowak R, Mulawka JJ, Borsuk P, Węgleński P, et al. DNA computing: implementation of data flow logical operations. Future Gener Comput Syst 2001;17:361–78.
15. Ran T, Kaplan S, Shapiro E. Molecular implementation of simple logic program. Nat Nanotechnol 2009;10:642–8.
16. Wąsiewicz P, Janczak, T, Mulawka, JJ, Płucienniczak A. The inference based on molecular computing. Cybernet Syst 2000;31:283–315.
17. Sainz de Murieta I, Rodriguez-Paton A. DNA biosensors that reason. Biosystems 2012;109:91–104.
18. Wąsiewicz P, Mulawka JJ. Molecular genetic programming. Soft Comput 2001;5:106–13.
19. Freund R, Kari L, Paun G. DNA computing based on splicing: the existence of universal computers. Theory Comput Syst 1999;32:69–112.
20. Winfree E. DNA computing by self-assembly. Bridge 2003;33: 31–8.
21. Paun G. Computing with membrane. J Comput Syst Sci 2000;61:108–43.
22. Kari L, Paun G, Rozenberg G, Salomaa A, Yu S. DNA computing, sticker systems, and universality, Acta Inform 1998;35:401–20.
23. Rothemund PW. Folding DNA to create nanoscale shapes and patterns. Nature 2006;440:297–302.
24. Gopinath A, Miyazono E, Faraon A, Rothemund PW. Engineering and mapping nanocavity emission via precision placement of DNA origami. Nature 2016;535:401–5.
25. Liu H, Wang J, Song S, Fan C, Gothelf KV. A DNA-based system for selecting and displaying the combined result of two input variables. Nat Commun 2015;10089:1–7.
26. Zhirnov V, Zadegan RM, Sandhu G, Church GM, Hughes W. Nucleic acid memory. Nat Mater 2016;15:366–70.
27. Rothemund PW. DNA and restriction enzyme implementation of Turing machines. Discrete Math Theor Comput Sci 1995;27:75–120.
28. Cavaliere M, Jonoska N, Yogev S, Piran R, Keinan E. Seeman N. Biomolecular implementation of computing devices with unbounded memory. Lect Notes Comput Sci 2005;3384:35–49.
29. Krasiński T, Sakowski S, Popławski T. Autonomous push-down automaton built on DNA. Informatica 2012;36:263–76.
30. Benenson Y, Paz-Elizur T, Adar R, Keinan E, Livneh Z, Shapiro E. Programmable and autonomous computing machine made of biomolecules. Nature 2001;414:430–4.
31. Hopcroft J, Motwani R, Ullman J. Introduction to automata theory, languages and computation. 3rd ed. Boston: Addison-Wesley, 2006.
32. Sipser M. Introduction to the theory of computation. 2nd ed. Boston: Thomson Course Technology, 2006.
33. 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 2003;100:2191–6.
34. Chen P, Jing L, Jian Z, Lin H, Zhizhou Z. Differential dependence on DNA ligase of type II restriction enzymes: a practical way toward ligase-free DNA automaton. Biochem Biophys Res Commun 2007;353:733–7.
35. Krasiński T, Sakowski S. Extended Shapiro finite state automaton. Found Comput Decis Sci 2008;33:241–55.
36. Krasiński T, Sakowski S, Waldmajer J, Popławski T. Arithmetical analysis of biomolecular finite automaton. Fundam Inform 2013;128:463–74.
37. Unold O, Troć M, Dobosz T, Trusiewicz A. Extended molecular computing model. WSEAS 2004;1:15–9.
38. Soreni M, Yogev S, Kossoy E, Shoham Y, Keinan E. Parallel biomolecular computation on surfaces with advanced finite automata. J Am Chem Soc 2005;127:3935–43.
39. Krasiński T, Sakowski S, Popławski T. Towards an autonomous multistate biomolecular devices built on DNA. World Congress on Nature and Biologically Inspired Computing, IEEE, 2014; 23–28.
40. Ukai H, Ukai-Tadenuma M, Ogiu T, Tsuji H. A new technique to prevent self-ligation of DNA. J Biotechnol 2002;97:233–42.
41. Benner S, Allemann R, Ellington A, Ge L, Glasfeld A, Leanz G, et al. Natural selection, protein engineering, and the last riboorganism: rational model building in biochemistry. Cold Spring Harb Symp Quant Biol 1987;52:53–63.
42. Henry A, Romesberg FE. Beyond A, C, G and T: augmenting nature's alphabet. Curr Opin Chem Biol 2003;7:727–33.

Supplemental Material: The online version of this article (DOI: 10.1515/znc-2016-0137) offers supplementary material, available to authorized users.