Biocybernetics and Biomedical Engineering
2008, Volume 28, Number 4, pp. 3–19

Finite State Automata Built on DNA

ROBERT NOWAK[1,*], ANDRZEJ PLUCIENNICZAK[2]

[1] *Warsaw University of Technology, Warsaw, Poland*
[2] *Institute of Biotechnology and Antibiotics, Warsaw, Poland*

This paper describes a non-deterministic finite-state automaton based on DNA strands. The automaton uses massive parallel processing offered by molecular approach for computation and exhibits a number of advantages over traditional electronic implementations. This device is used to analyze DNA molecules, whether they are described by specified regular expression. Presented ideas are confirmed by experiment performed in a genetic engineering laboratory.

K e y w o r d s: DNA computing, automata, regular expressions, molecular computing

1. Introduction

The paper concerns moleclar computations i.e. new discipline, which uses molecules to performing calculations [1, 2]. The described method uses DNA (deoxyribonucleic acid) molecules for building finite state automata. The double helix of DNA is formed from two separate DNA strands, connected together (head-to-toe) by hydrogen bonds. The DNA strands may be viewed as a chain of nucleotides. There are four nucleotides: adenine, cytosine, guanine, and thymine, abbreviated to `A`, `C`, `G`, and `T` respectively. Each strand has a natural orientation, denoted (according to chemical convention) as 5' and 3' end. The hydrogen bond is selective, `A` bonds with `T`, and `G` bonds with `C`, the pairs `(A,T)` and `(G,C)` are complementary. The DNA strands are complementary if they are built from complementary nucleotides. More information about DNA and basic operations (i.e. hybridization, denaturation, ligation, cutting, PCR) from computer scientists point of view can be found in [3–5].

An ***alphabet*** is a finite nonempty set of symbols. ***Symbol*** over some alphabet Σ, denoted in this paper by a small letter or digit, is represented by a sequence

* Correspondence to: Robert Nowak, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland, e-mail: r.m.nowak@elka.pw.edu.pl
Received 19 February 2008; Accepted 10 June 2008

of consecutive nucleotides of length n, $\overline{x}$ denotes the sequence complementary to the sequence representing symbol x, e.g. if x is represented by `ATCCCA` (sequences are written from `5'` to `3'` end), the complementary sequence is `3'-TAGGGT-5'`, thus $\overline{x}$ is `TGGGAT`. A ***string*** over given alphabet is any finite sequence of symbols (e.g. ε, a, b, aa, ab, aab, are the strings over $\Sigma=\{a,b\}$, ε represents empty string). The length of a string R (the number of symbol occurrences in R) is denoted by $|R|$, e.g. $|aab|=3$, $|\varepsilon|=0$. The strings are represented by DNA strands, and denoted in this paper by capital letter. For example, consider the alphabet $\Sigma=\{a,b\}$, where symbols are represented by `ATCCCA`, `GGTCCT` respectively. The DNA strand for $R=abb$ has sequence `ATCCCAGGTCCTGGTCCT`.

A set of strings over given alphabet is called a ***formal language***. A ***formal grammar*** is a precise description of a formal language. The ***regular languages*** are the simplest class in the Chomsky hierarchy of formal grammars. The ***right-linear grammar*** is quadruple $\mathbf{G}=(\mathbf{N},\mathbf{T},\mathbf{P},q0)$, where $\mathbf{N}$ is a nonterminal alphabet, $\mathbf{T}$ is a terminal alphabet $\mathbf{T}\subseteq\Sigma$, q_0 is the starting symbol (axiom) $q_0\in\mathbf{N}$, and $\mathbf{P}$ is the set of production rules. Production rules conform the pattern $1\rightarrow a2$ or $1\rightarrow a$, where $1\in\mathbf{N}$, $2\in\mathbf{N}$, and $a\in\mathbf{T}$. In this work numbers denote nonterminal symbols, letters denote terminal symbols. In practise the ***regular expressions*** are commonly used to describe regular language. For any regular expression an equivalent right-linear grammar can be constructed. More information about languages and grammars can be found in [6].

Decision whether a given string belongs to a given regular language is undertaken by a ***finite state automaton***. The finite state automaton can be constructed [6] for any regular language. If the length of regular expression describing given regular language is $|R|$, then simple algorithm (of linear time complexity for electronic computer) can construct a non-deterministic finite state automaton. The number of states (memory complexity) is $O(|R|)$, and time complexity to analyze string X is $O(|R|*|X|)$. The deterministic finite state automaton has the number of states exponentially dependent on length of regular expression, so memory complexity is $O(2^{|R|})$, and the analysis for string X takes $O(|X|)$ steps. A sample finite state automaton is depicted in Fig. 1.

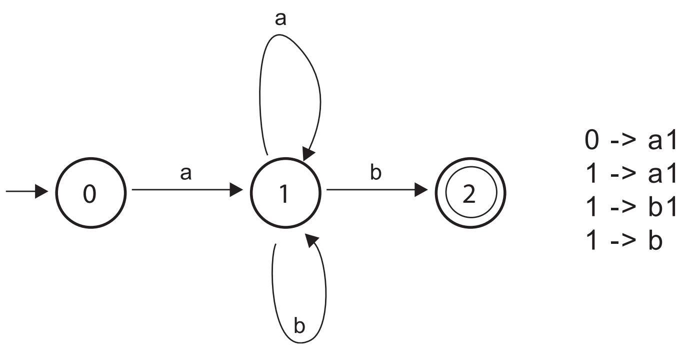


Fig. 1. Finite state automaton and the production rules for the language $a(a|b)^*b$

The automaton recognizes strings belonging to the language $a(a|b)^*b$. The grammar $\mathbf{G}=(\mathbf{N},\mathbf{T},\mathbf{P},q0)$, where $\mathbf{N}=\{0,1\}$, $\mathbf{T}=\{a,b\}$, $q0=0$, and **P** showed in Fig. 1 generates strings belonging to the language.

The idea described in this paper is to build a non-deterministic finite state automaton *in vitro*. Such a device uses massive parallelism given by molecular approach, and has size complexity (understood as a number of different molecules) $O(|R|)$, where R is regular expression describing given language. The analysis for string X has $O(|X|)$ time complexity.

2. Molecular Production

The ***molecular production*** is a biological system which conditionally creates designed DNA strand. It is the basic element in a molecular automaton used to implement transition.

The molecular production, denoted $A \rightarrow B$, creates string XB if and only if the input is XA (A, B, X are sub-strings, $|A|>0$). Such system checks if the sequence of nucleotides representing condition (here A) is presented at the 3' end of the input string, and then creates the output string: the DNA strand is copied from the input, but the condition sequence is replaced by the sequence representing the result of production (here B). Therefore, for the input XA, the XB is obtained (Fig. 2). It should be mentioned, that XA is also presented in the output, because the input and the output are not separated.

If the strand representing the input string has not the condition sequence at the 3' end, the molecular production creates nothing. For example production $A \rightarrow B$ for input XC, where $C=A$, provides only XC (Fig. 2).

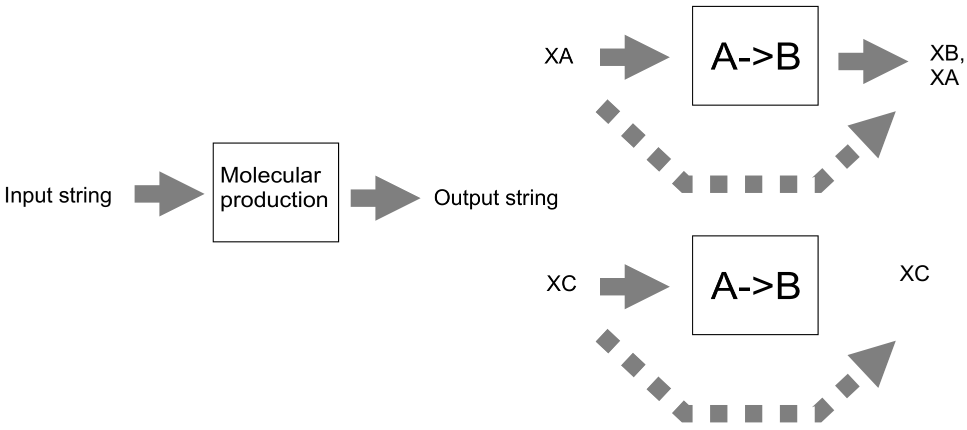


Fig. 2. Molecular production $A \rightarrow B$; X, A, B, C are strings, and $|A|>0$, $A=C$

3. Molecular Finite State Automaton

3.1. Reductions for Right-linear Grammars

Theorem:
If string S belongs to a language generated by the right-linear grammar $\mathbf{G}=(\mathbf{N},\mathbf{T},\mathbf{P},q0)$, then it can be reduced to the string $q0$ (axiom). The following reduction rules are used for the last symbols in a string:

- $a \rightarrow 0$ when $0 \rightarrow a \in \mathbf{P}$,
- $a1 \rightarrow 0$ when $0 \rightarrow a1 \in \mathbf{P}$.

Lemma:
When string is generated from the axiom ($q0$) by the right-linear grammar it has at most one non-terminal symbol. This is the last symbol of the string.

Proof of lemma (mathematical induction):
Assume, that $S_n=w_1w_2...w_n A_n$, where $w_i \in T$, $A_n \in N$. S_{n+1} should be obtained from S_n by production $A_n \rightarrow w_{n+1}A_{n+1}$ (it retain the condition) or by production $A_n \rightarrow w_{n+1}$ (also retains it).

Generated strings are: $q_0 \rightarrow w_1A_1 \rightarrow w_1w_2A_2 \rightarrow ... \rightarrow w_1w_2...w_{n-1}A_{n-1} \rightarrow w_1w_2...w_n$, where $w_i \in T$, $A_i \in N$.

Proof of theorem:
String $S_n=w_1w_2...w_n$ can be generated by right-linear grammar $\mathbf{G}=(\mathbf{N},\mathbf{T},\mathbf{P},q_0)$, only from string $S_{n-1}=w_1w_2...w_{n-1}A_{n-1}$, where $A_{n-1} \rightarrow w_n \in \mathbf{P}$. Therefore strings S_{n-1} can be constructed from S_n by reductions $w_n \rightarrow A_{n-1}$. If $\mathbf{P}$ does not contain production rule $A_{n-1} \rightarrow w_n$, then string $w_1w_2...w_n$ does not belong to language generated by **G.**

String $S_n=w_1w_2...w_n A_n$ can be created only from strings $w_1w_2...w_{n-1}A_{n-1}$, where $A_{n-1} \rightarrow w_n A_n \in \mathbf{P}$. If production rule $A_{n-1} \rightarrow w_n A_n \in \mathbf{P}$, then string $Sn=w_1w_2...w_n A_n$ does not belong to language generated by **G**. Strings S_{n-1} are constructed from S_n by reductions $w_n A_n \rightarrow A_{n-1}$. Because $|S_{n-1}|=|S_n|-1$, after $n-1$ steps, the set of strings of length equal 1 is obtained. If the axiom (q_0) is in this set, the string $w_1w_2...w_n A_n$ can be generated by **G**.

3.2. Molecular Automaton Based on Reductions

For a given language the corresponding right-linear grammar is constructed. To analyze the input string the reductions (described in the theorem) are performed. If the axiom (starting symbol) is obtained the input string belongs to the grammar (is accepted).

Such an idea is the core of the molecular automation. This device takes advantage of molecular production to implement reductions. The algorithm is shown in Fig. 3.

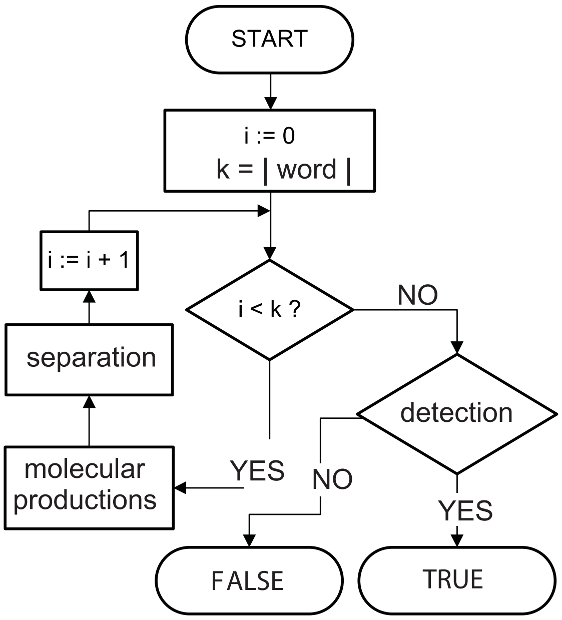


Fig. 3. Molecular automaton – algorithm

Firstly, the automaton is prepared and DNA strand representing the input string is added to a vessel.

Then k steps (where $k=|S|$) of productions and separations are performed. After each production the last symbols from the string could be reduced. The separation removes the input string (present in the output of the molecular production), i.e. for input XA and molecular production $A\rightarrow B$ only XB is kept. It should be noted that each molecular production could work independently of each other, so in a single step many different reductions should be performed.

Finally the axiom is detected. If such string is obtained, the answer is *true*, the input string is accepted by automaton. Otherwise, the answer is *false*.

3.3. Example

Consider the regular language $a(a|b)^*b$ and the right-linear grammar shown in Fig. 1. The reductions (molecular productions) for this language are: $b\rightarrow 1$, $b1\rightarrow 1$, $a1\rightarrow 1$ and $a1\rightarrow 0$.

The molecular automaton performs 3 steps when the input string *abb* is analyzed: $abb\rightarrow ab1\rightarrow a1\rightarrow \{0,1\}$. The axiom (symbol 0) is present, thus *abb* belongs to the given language.

For *bbb* and the same automaton the reductions are: $bbb \rightarrow bb1 \rightarrow b1 \rightarrow 1$. The axiom is not present, so *bbb* is rejected.

4. Molecular Production – Realization

The molecular production is the basic element used to build the automaton considered in this paper. Below the details of this process are presented.

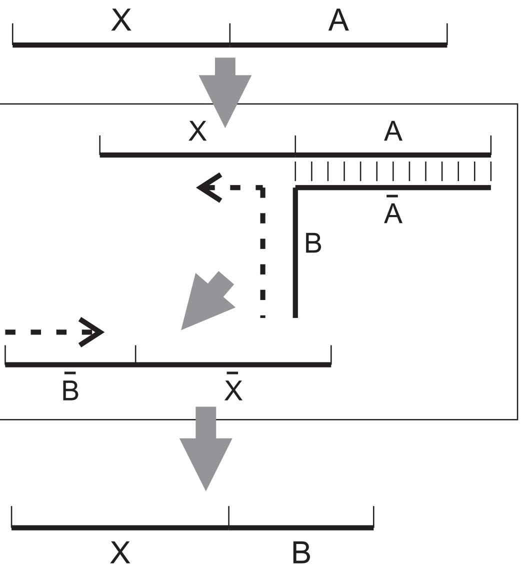


Fig. 4. Molecular production $A \rightarrow B$. The production engine has sequence ĀB

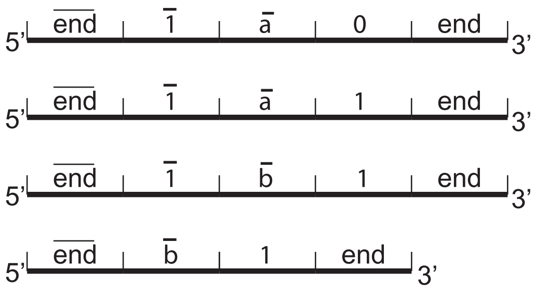
The process of the molecular production (illustrated in Fig. 4) needs the DNA strand called a production engine. This strand contains two parts: the first is complementary to the conditional part of the molecular production (i.e. Ā *for* $A \rightarrow B$), the second has nucleotides representing the result of production (i.e. B for $A \rightarrow B$).

Firstly, the production engine partially hybridizes to the input string (only if it has the proper sequence on 3' end). Next, the special polymerization with DNA polymerase "jump" is performed. A strand built by polymerase has the sequence complementary to the output string. Finally, the polymerization is performed, so the output string is produced. It should be noted that if the hybridization does not occur (the production engine and the input string are not partially complementary), polymerase does produce only a strand complementary to the production engine, which can be easily removed.

The probability of DNA polymerase "jump" is very small, so in experiments the PCR is applied. PCR (and dilution) can also remove the strands representing input strings (separation in Fig. 3).

5. Molecular Automaton Example

The molecular automaton (described in section 3) is represented in vessel by a set of production engines. For the language $a(a|b)^*b$ these molecules are depicted in Fig. 5. For the considered language there are 4 productions, the sequences correspond to reductions: $a1 \rightarrow 0$, $a1 \rightarrow 1$, $b1 \rightarrow 1$ and $b \rightarrow 1$ respectively.



5' $\overline{\text{end}}$ $\overline{1}$ $\overline{a}$ 0 end 3'
5' $\overline{\text{end}}$ $\overline{1}$ $\overline{a}$ 1 end 3'
5' $\overline{\text{end}}$ $\overline{1}$ $\overline{b}$ 1 end 3'
5' $\overline{\text{end}}$ $\overline{b}$ 1 end 3'

Fig. 5. The molecular automaton (molecular engines) for the language $a(a|b)^*b$

The DNA strand representing a model string *aab* is shown in Fig. 6. As denoted previously there are sequences representing symbols, and some temporary ones, denoted *start*, *end,* used as primers.

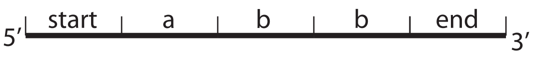


Fig. 6. DNA strand representing a model word *abb*

At the beginning of calculations the molecules implementing the automaton (production engines) and input string are put into the vessel. Next the molecular productions are performed (firstly the hybridization, next polymerization with “jump”, then denaturation and finally polymerization, like in section 4). Because of the length of string *aab*, three steps of the molecular production and separation are performed.

The first step is schematically depicted in Fig. 7. The production engine representing reduction $b \rightarrow 1$ bonds to the strand working as the input string (the reduction $b \rightarrow 1$ is called to be active). The other engines are not bonded, so after the molecular production and separation in the vessel the molecule corresponding to $aa1$ string were presented.

In the second step, depicted in Fig. 8, the reduction $a1 \rightarrow 1$ is active, and the string $a1$ is obtained.

The third step (Fig. 9) shows parallelism of the described approach. The two reductions are active: $a1 \rightarrow 0$ and $a1 \rightarrow 1$. So two different molecules were created: molecules representing string 0 and 1 respectively.

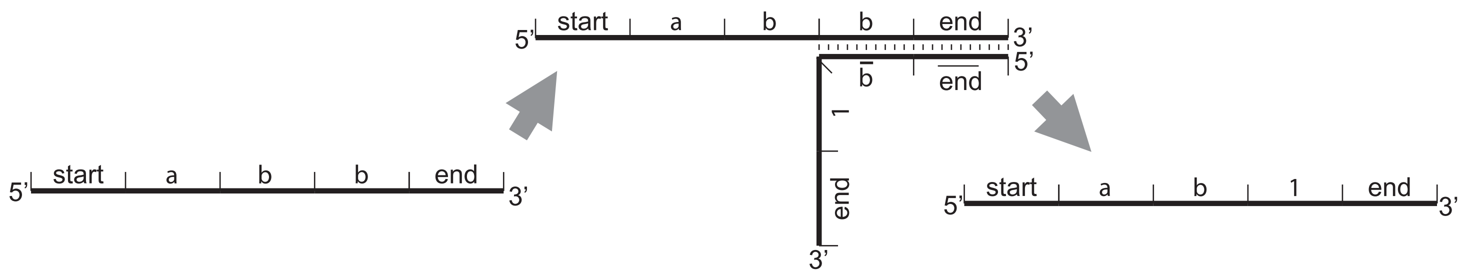


Fig. 7. Examine string *aab* (for the language *a*(*a*|*b*)**b*). Step 1.

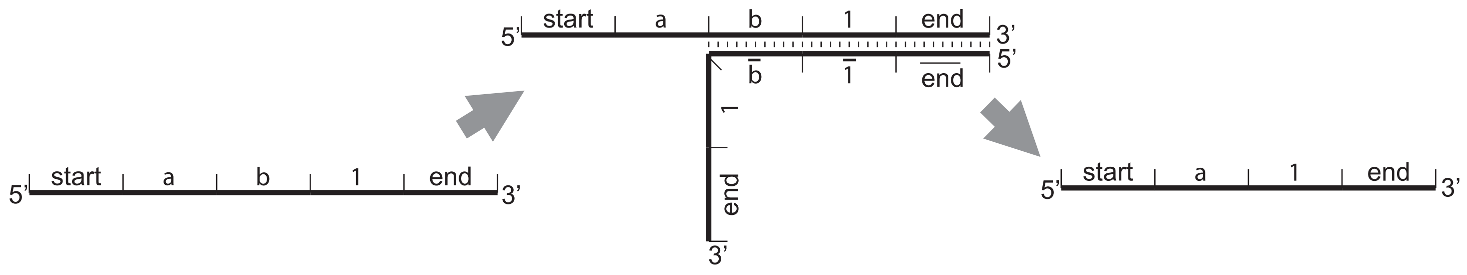


Fig. 8. Examine string *aab* (for the language *a*(*a*|*b*)**b*). Step 2

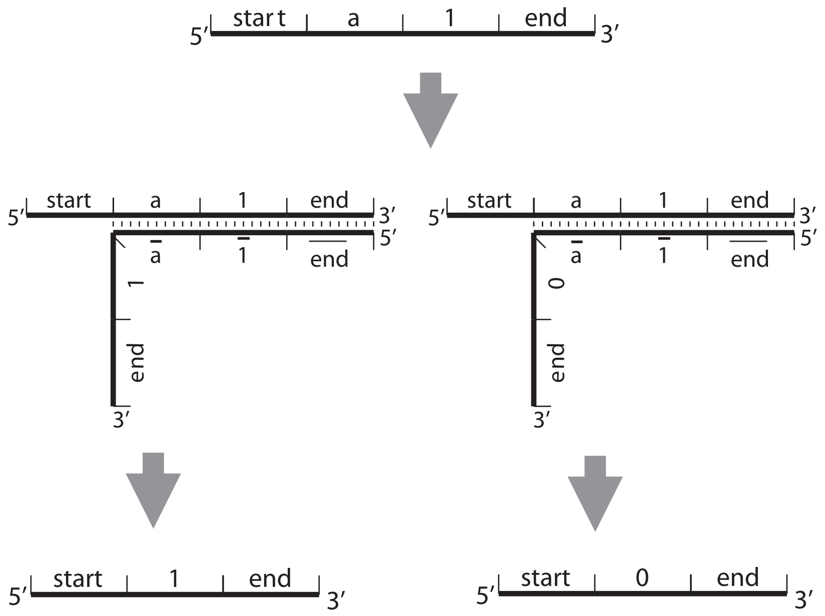


Fig. 9. Examine string *aab* (for the language *a*(*a*|*b*)**b*). Step 3

Finally, the detection is performed. It is done by checking if the molecule representing the axiom (string 0 here) is presented in the vessel. Because such DNA strand is produced in the third step, then answer is true. The string *aab* is accepted by the molecular automaton.

6. Experimental Verification

The experiments realized in a genetic engineering laboratory confirmed the presented ideas. The process of the molecular production was performed and the possibility of building of the molecular automata using many of them was verified. The automata built in the laboratory were very simple, as depicted in Fig. 10.

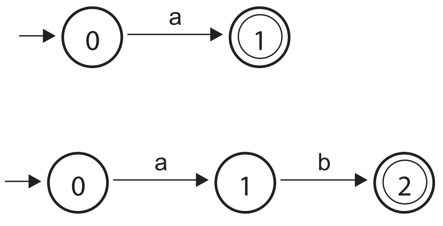


Fig. 10. The finite automata build in the laboratory

6.1. Verification of Molecular Production

The experiment was performed as described in section 4, details supplied below. The single stranded DNA molecule from *M*13 fag, described as *m*13, (genBank: www.ncbi.nlm.nih.gov; No X02513, produced by Amersham) was applied as the input string. The production engine called *m*13*HAK* was synthetic molecule having sequence provided in Table 1.

Table 1. Sequence of DNA molecules used in the experiment

Name	Length	Sequence
*m*13	7249	`M13mp18` – GenBank
*m*13*PKO*	21	`CTA GCA CTA CAA CTC GGA CTA`
*m*13*HAK*	55	`GAG GTC ATT TTT GCG GAT GGC TTA GAG CTT CCG -`
		`GTA GTC CGA GTT GTA GTG CTA G`
*m*13*P*	22	`CTA TTA GTA GAA TTG ATG CCA C`

The hybridization was performed on the mixture showed in Table 2, which was heated to 95°C (denaturation all molecules) and kept in 72°C for 1 minute. As a result, the structure illustrated in Fig. 11 was created. The molecule m13HAK sticked to m13 (sequences partly complement) and m13PKO (used as primer) hybridized to m13HAK.

Table 2. Mixture used for hybridization in verification of the molecular production

Name	Concentration	Amount
*m*13	0.44 pM/µl	5 µl
*m*13*HAK*	5.0 pM/µl	1 µl
*m*13*PKO*	5.0 pM/µl	1 µl
Buffer	10×	4 µl
H2O		23 µl
TOTAL		34 µl

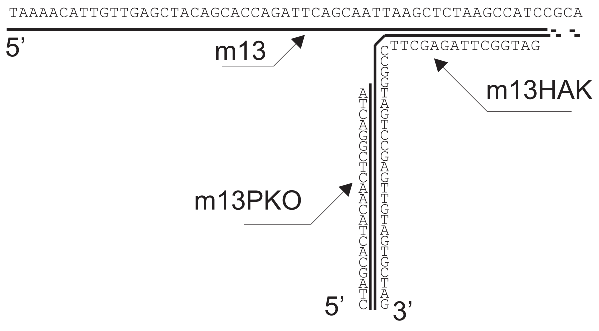

TAAAACATTGTTGAGCTACAGCACCAGATTCAGCAATTAAGCTCTAAGCCATCCGCA
5’
m13
TTCGAGATTCGGTAG
m13HAK
CCGGTAGTCCGAGTTGTAGTGCTAG
ATCAGGCTCAACATCACGATC
m13PKO
5’
3’

Fig. 11. DNA strands created after hybridization

Next the polymerization with DNA polymerase “jump” was performed. To mixture kept in 50°C the 5 μl of dNTP (20 μM/μl, Amersham) and 1 μl Taq DNA polymerase (2u/μl, produced by Institute of Biotechnology and Antibiotics from *Thermophilus Aquatius*) was added.

After 20 minutes the mixture was treated with phenol and chloroform-isoamyl alcohol mixture, amplified by PCR with primers m13P and m13PKO (26 and 29 cycles: 94°C 15sec, 50°C 15sec, 72°C 30sec). 10 μl of the PCR products were electrophoresed on 6% polyacrylamide gels (acrylamide: bisacrylamide = 59:1) with TAE buffer, stained in ethidium bromide at 0.5 μg/L aqueous solution for 10 min. Image of the gel (Fig. 12) was made using a White/Ultraviolet Transiluminator.

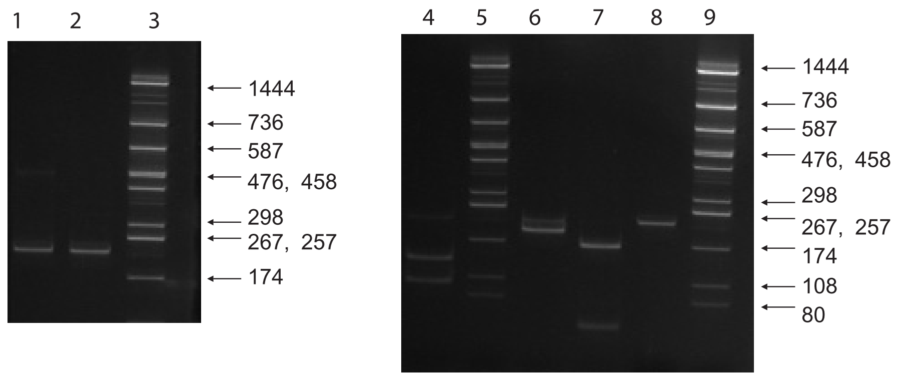


Fig. 12. Gel image after the DNA polymerase “jump” reaction: strip 1,2 and 8: products of PCR (DNA before cutting); strip 3,5 and 9: pattern (1444, 736, 587, 476, 458, 434, 298, 267, 257, 174, 102, 80, 30); strip 4: *Hinf*I; strip 6: *Hpa*II; strip 7: *RsaI*

The molecule had proper length, however additional verification was performed. The products of PCR were cut by restrictazes *Rsa*I, *Hinf*I, *Hpa*II (Amersham). Strips 4, 6 and 7 in Fig. 12 proved that the molecules had proper sequences.

The estimated probability of DNA polymerase “jump” is about $3*10^{-8}$. This value was obtained comparing the brightness of the strips.

6.2. Verification of Simple Molecular Automaton

The two state automaton, depicted in Fig. 10 containing two molecular productions: $a \rightarrow b$ and $b \rightarrow c$ was practically tested in the laboratory. In the experiment the input string *xa* was transformed into *xb* and then into *xc*. The simplified schema of this experiment is shown in Fig. 13. The idea was described in section 3 and section 5, details supplied below.

The sequences of DNA molecules used in this experiments are showed in Table 3. The molecule called *m*13 represented the input string (*xa*), the *aut* represented the string *xb*, the production engine *a*→*b* was *m*13*HAK*2, the *m*13*HAK*3 was the production engine *b*→*c*, the molecule *m*13*Z* was a support for molecular production and others (*m*13*PKP*, *m*13*P*, *m*13*PKP2)* were used as primers.

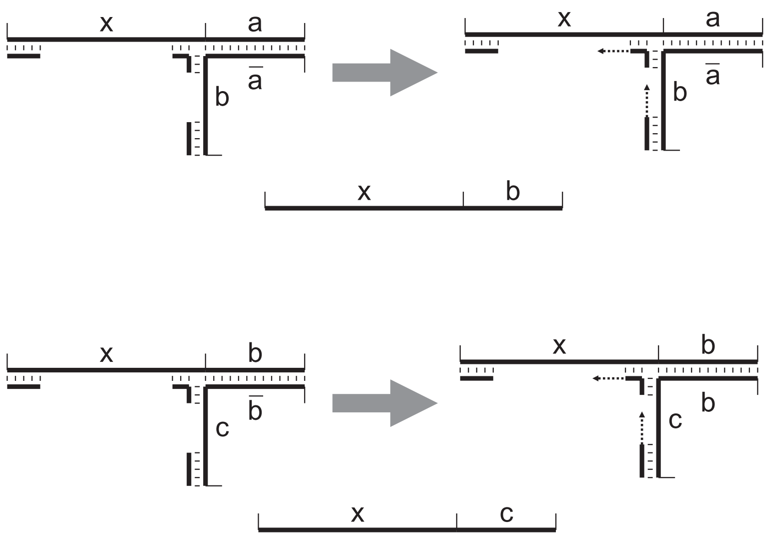


Fig. 13. Experiment for the finite state automaton. a) the molecular production $a \to b$ construct the *xb* string where *xa* is the input one, b) the second production $b \to c$ makes *xc* from *xb*

Table 3. Sequence of DNA molecules used in the experiment

Name	Length	Sequence
*m*13	7249	M13mp18 – GenBank
*m*13*HAK*2	60	ATC TGG TGC TGT AGC TCA ACA TGT TCC GGA -
		GCA CCA GAT ATC TTC GAG TTG TAG TGC TAG
*m*13*Z*	25	TGC TCC GGT TAA ATA TGC AAC TAA A
*m*13*PKP*	21	CTA CGA CTA CAA CTC GGA GAT
*m*13*P*	22	CTA TTA GTA GAA TTG ATG CCA C
aut	218	the string created by molecular production *a*→*b*
*m*13*HAK*3	56	CGA AGA TAT CTG GTG CTC CGG CCG GAG CAC -
		GAG AAT TCC ATA GGA CCT TGC GCT CC
*m*13*PKP*2	21	GGA GCG CAA GGT CCT ATG GAA

Firstly the molecules *m*13*Z* were kinazed (T4 kinaze) at 37°*C* for 1hour, than the enzyme was thermally deactivated. Secondly the molecular production $a \to b$ was performed, next the production $b \to c$ was done, finally the molecule representing output string was checked. This process was described in section 3 from computer science point of view.

The molecular production $a \rightarrow b$ was performed in a different way than described in section 4. The respective steps were:

1. Hybridization: the mixture showed in Fig. 14 was heated to 95°*C* for 1 minute (denaturation of all molecules) and kept at 72°C for 1 minute. The structure illustrated in Fig. 14 was created. The molecule *m*13*Z* sticked to both DNA strands, eliminating the DNA polymerase "jump" and caused that molecular production was much more efficient.

2. The polymerization: to the vessel containing the products of hybridization 0.5 µl Klenov polimeraze (5 u/µl) were added, and the polymerization was performed at 20°C for 2 minutes. There were two starters: *m*13*PKP* and *m*13*Z* (Table 4, Fig. 14). The enzyme was thermally deactivated (the mixture was heated to 65°C).

3. The ligation: 10 µl of mixture after polymerization, 1 µl ligase enzyme (Amersham) and ligase buffer were kept at 16°C for 24hours.

4. PCR: the products of ligation were amplified by PCR with primers m13P and m13PKP (cycles: 94°C 15sec, 50°C 15sec, 72°C 30sec).

5. Check: 10 µl of the PCR products were electrophoresed on 6% polyacrylamide gels (acrylamide: bisacrylamide = 59:1) with TAE buffer and stained in ethidium bromide at 0.5 µg/L aqueous solution for 10 min. Image of the gel (Fig. 15) made by using a White/Ultraviolet Transiluminator proved the proper length of the aut molecule.

6. Check: 10 µl of the PCR products were treated with phenol and chloroform-isoamyl alcohol mixture, and were cut by restrictazes *EcoR*V, *Hinf*I and *Msp*I (Amersham) to provide that molecules have proper sequences. The gel image is shown in Fig. 16.

It should be mentioned, that PCR(step 4) performed the separation (Fig. 3). This

Table 4. Mixture for verification of the molecular production $a \rightarrow b$

Name	Concentration	Amount
m13	0.1 pM/µl	2.0 µl
m13HAK2	6 pM/µl	1.0 µl
m13Z	5 pM/µl	1.5 µl
m13PKP	8 pM/µl	1.0 µl
dNTP	20 µM/µl	2.0 µl
Buffer	10×	2.0 µl
H2O		10.0 µl
TOTAL		19.5 µl

process produced the *aut* molecule of length 218*bp* and concentration 0.5 pM/µl (10 ng/µl).

The molecular production $b \rightarrow c$ is performed similarly. The PCR products of the

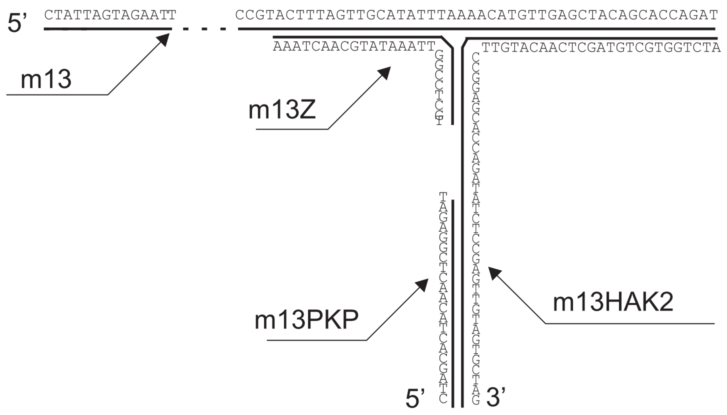

5'
CTATTAGTAGAATT
CCGTACTTTAGTTGCATATTTAAAACATGTTGAGCTACAGCACCAGAT
AAATCAACGTATAAATT
TTGTACAACTCGATGTCGTGGTCTA
m13
m13Z
GGCCTCGT
CCGGAGCACCAGATATCTCCGAGTTGTAGTGCTAG
TAGAGGCTCAACATCACGATC
m13PKP
m13HAK2
5'
3'

Fig. 14. DNA molecules used to verification of the molecular production $a \rightarrow b$

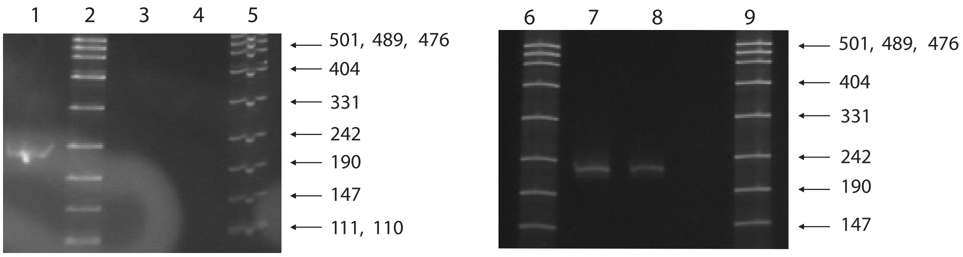


Fig. 15. Molecules after $a \rightarrow b$ molecular production: strip 1 and 7: PCR after 23 cycles, strip 2, 5, 6 and 9: pattern (1444, 736, 587, 476, 458, 434, 298, 267, 257, 174, 102, 80, 30); strip 3: PCR 19 cycles; strip 4: PCR 15 cycles; strip 8: PCR 24 cycles

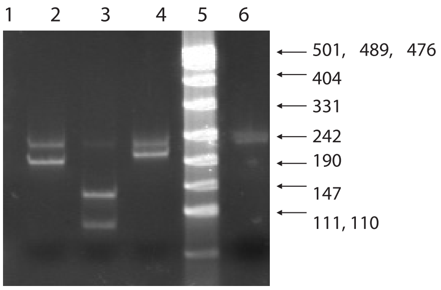


Fig. 16. Gel image after cutting *aut* molecule; strip 1: *Msp*I, strip 2: *Hinf*I, strip 3: *EcoR*V, strip 4: pattern (1444, 736, 587, 476, 458, 434, 298, 267, 257, 174, 102, 80, 30), strip 5: DNA before cutting

production $a \rightarrow b$, containing the *aut* molecule were used in: hybridization, polimerisation, ligation and amplification and then the molecule representing *xc* string was obtained, details supplied below.

The hybridization mixture (Table 5) was heated to 95°C and kept at 72°C.

The structure showed in Fig. 17 was created. Next polimerisation and ligation was performed. Finally, the molecules were amplified by PCR with primers m13P and m13PKP2, electrophoresed and photographed (Fig. 18). The products of PCR were cut by restriction enzymes to prove having proper sequences.
The experiment manifested, that many consecutive molecular can perform computa-

Table 5. Mixture for verification of the molecular production $a \to b$

Name	Concentration	Amount
aut	0.5 pM/µl	4 µl
*m*13*HAK*3	10 pM/µl	2 µl
*m*13*Z*	5 pM/µl	3 µl
*m*13*PKP*2	10 pM/µl	2 µl
dNTP	20 µM/µl	2 µl
Buffer	10×	3 µl
H2O		13 µl
TOTAL		29 µl

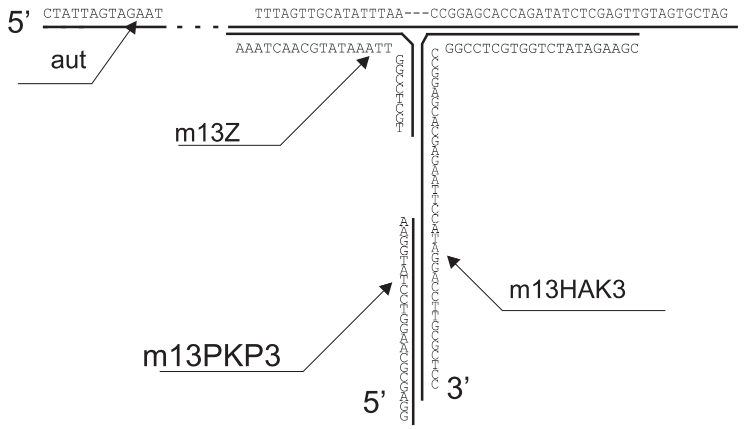


Fig. 17. Structure after hybridization when reaction to verify the molecular production $b \to c$ was performed

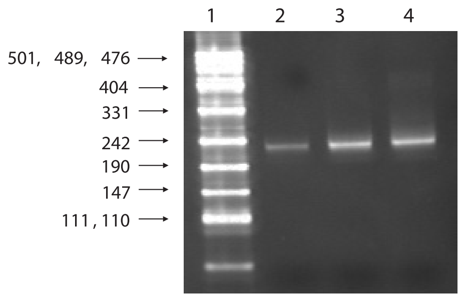


Fig. 18. Gel image after the production $b \to c$. strip 1: pattern (1444, 736, 501, 489, 476, 404, 331, 242, 190, 147, 111, 110); strip 2: PCR after 23 cycles; strip 3: PCR after 26 cycles; strip 4: PCR after 29 cycles

tions, thus it was demonstrated that molecular automata described in this work can be practically build and used.

7. Conclusion

The short comparison between complexity of different finite state automata is presented in Table 6. The molecular approach has advantages over electronic implementations, because each possible transitions from a given state are simultaneously considered (take advantage of massive parallel processing).

There are a few others works describing realization of automata by using

Table 6. Complexity of finite state automata described by the regular expression r when the string S is analyzed. Size for the molecular automaton is the number of different molecules used for calculation

Automaton	Size	Time
Electronic nondeterministic	$O(\lvert\mathbf{r}\rvert)$	$O(\lvert\mathbf{r}\rvert*\lvert S\rvert)$
Electronic deterministic	$O(2\lvert\mathbf{r}\rvert)$	$O(\lvert S\rvert)$
Molecular	$O(\lvert\mathbf{r}\rvert)$	$O(\lvert S\rvert)$

the molecular approach. In [2] only propositions are given, in [7] the human assistance is needed. The described method employs a single vessel to code many states (because it implements a non-deterministic automaton), and the person reads the current symbol from the input string, and decides to which vessel the molecules should be added (simulating transition). It complicates the experiments and makes the process slower, more expensive and much prone to errors. The interesting idea shown in [8], which uses *FokI* enzyme, was experimentally proved. The main disadvantage of this method is small maximum number of states and transitions (256).

The presented non-deterministic finite state automaton can be treated as an alternative way of performing molecular computation. It is the step toward constructing a molecular computer.

Practically, it might be used in biological and medical research, for searching DNA sequences described by the regular expression. When a requested sequence is simple (can be described by the regular expression), the described non-deterministic automaton perform this task quickly and inexpensively (when compared with currently used DNA sequencing), so e.g. the diagnosis of a genetic disease can be performed on a large scale.

References

1. Amos M.: Theoretical and experimental DNA computation. Springer, 2005.
2. Păun G., Rozenberg G., Salomaa A.: DNA Computing: NewComputing Paradigms. Springer, 1998.
3. Baxevanis A.D., Ouellette B.F.: Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, John Wiley & Sons, Inc., 2001.
4. Deonier R.C., Tavare S., Waterman M.S.: Computational Genome Analysis: An Introduction Springer, 2005.
5. Lila K., Kitto R., Gloor G.: A computer scientist's guide to molecular biology. Soft Computing, 2001, 5, 95–101.
6. Hopcroft J., Ullman J.: Introduction to Automata Theory, Languages, and Computation. Addison Wesley, 1979.
7. Rose J., Gao Y., Garzon M., Murphy R.C., Deaton R., Franceschetti S.: Dna implementation of finite-state machines. 2nd Anneal Genetic Programming Conference, Morgen Kaufmann, 1997, 160–165.
8. Benenson Y., Paz-Elizur T., Adar R., Keinan E., Livneh Z., Shapiro E.: Programmable and autonomous computing machine made of biomolecules. Nature, 2001, 414, 430–434.