Development of an *In Vivo* Computer Based on *Escherichia coli*

Hirotaka Nakagawa[1], Kensaku Sakamoto[2], and Yasubumi Sakakibara[1]

[1] Keio University, Department of Biosciences and Informatics,
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522, Japan
yasu@bio.keio.ac.jp
[2] RIKEN Genomic Sciences Center,
1-7-22 Suehiro-cho, Tsurumi, Yokohama, 230-0045, Japan
sakamoto@gsc.riken.jp

Abstract. We present a novel framework to develop a programmable and autonomous *in vivo* computer using *E. coli*, and implement *in vivo* finite-state automata based on the framework by employing the protein-synthesis mechanism of *E. coli*. Our fundamental idea to develop a programmable and autonomous finite-state automata on *E. coli* is that we first encode an input string into one plasmid, encode state-transition functions into the other plasmid, and introduce those two plasmids into an *E. coli* cell by electroporation. Second, we execute a protein-synthesis process in *E. coli* combined with four-base codon techniques to simulate a computation (accepting) process of finite automata, which has been proposed for *in vitro* translation-based computations in [8]. This approach enables us to develop a programmable *in vivo* computer by simply replacing a plasmid encoding a state-transition function with others. Further, our *in vivo* finite automata are autonomous because the protein-synthesis process is autonomously executed in the living *E. coli* cell. We show some successful experiments to run an *in vivo* finite-state automaton on *E. coli*.

1 Introduction

The finite-state automata (machines) are the most basic computational model in Chomsky hierarchy and are the start point to build universal DNA computers. Several works have attempted to develop finite automata *in vitro*. However, there have been no experimental research works which attempt to build a finite automaton *in vivo*.

We have previously proposed a method using the protein-synthesis mechanism combined with four-base codon techniques to simulate a computation (accepting) process of finite automata *in vitro* [8] (a codon is normally a triplet of base, and different base triplets encode different amino acids in protein). The proposed method is quite promising and has several advanced features such as the protein-synthesis process is very accurate and overcomes mis-hybridization problem in the self-assembly computation and further offers an autonomous computation. Our aim was to extend this novel principle into a living system, by employing the *in vivo* protein-synthesis mechanism of *E. coli*. This *in vivo* computation

A. Carbone and N.A. Pierce (Eds.): DNA11, LNCS 3892, pp. 203–212, 2006.

possesses the following two novel features, not found in any previous biomolecular computer. First, an *in vivo* finite automaton is implemented in a living *E. coli* cell; it does not mean that it is executed simply by an incubation at a certain temperature. Second, this automaton increases in number very rapidly according to the bacterial growth; one bacterial cell can multiply to over a million cells overnight. The present study explores the feasibility of *in vivo* computation.

The main feature of our *in vivo* computer based on *E. coli* is that we first encode an input string into one plasmid, encode state-transition functions into the other plasmid, and transform *E. coli* cells with these two plasmids by electroporation. Second, we execute a protein-synthesis process in *E. coli* combined with four-base codon techniques to simulate a computation (accepting) process of finite automata, which has been proposed for *in vitro* translation-based computations in [8]. The successful computations are detected by observing the expressions of a reporter gene linked to mRNA encoding an input data. Therefore, when an encoded finite automaton accepts an encoded input string, the reporter gene, *lacZ*, is expressed and hence we observe a blue color. When the automaton rejects the input string, the reporter gene is not expressed and hence we observe no blue color. Our *in vivo* computer system based on *E. coli* is illustrated in Fig. 1.

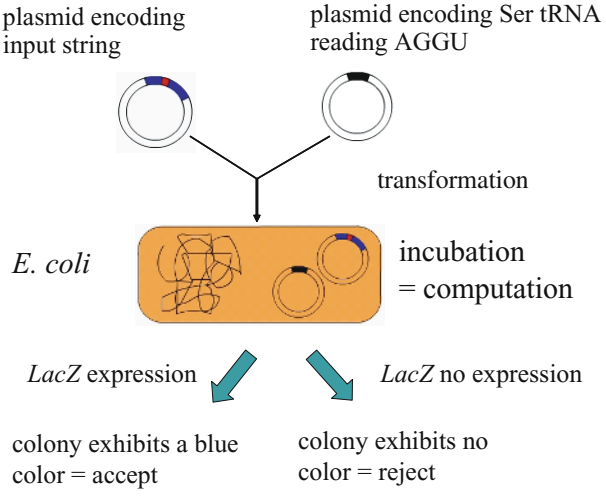

Fig. 1. The framework of our *in vivo* computer system based on *E. coli*

Thus, our *E. coli*-based computer enables us to develop a programmable and autonomous computer. To our knowledge, this is the first experimental development of *in vivo* computer and has succeeded to execute an finite-state automaton on *E. coli*.

2 Methods

2.1 A Framework of Programmable and Autonomous *In Vivo* Computer on *E. coli*

Two important issues on developing DNA-based computers are *programmable* and *autonomous*. We realize these two mechanisms by using the main features of our *in vivo* computer based on *E. coli*.

Programmable: The programmable means that a program is stored as a data (i.e., stored program computer) and any computation can be accomplished by just choosing a stored program. In DNA-based computers, it requires that a program is encoded into a molecule different from the main and fixed units of DNA computer, a molecule encoding programs can be stored and changed, and a change of molecules encoding programs accomplishes any computations.

The main features of our *in vivo* computer enable us to develop a programmable *in vivo* computer. We simply replace a plasmid encoding a state-transition function with other plasmid encoding a different state-transition function, and the *E. coli* cell transformed a new plasmid computes a different finite automaton.

Autonomous: The autonomous DNA computers mean that once we set a program and an input data and start a computation, the entire computational process is carried out without any operations from the outside. Our *in vivo* finite automata are autonomous in the sense that the protein-synthesis process which corresponds to a computation (accepting) process of an encoded finite automata is autonomously executed in a living *E. coli* cell and require no laboratory operations from the outside.

2.2 Simulating Computation Process of Finite Automata Using Four-Base Codons and Protein-Synthesis Mechanism

Sakakibara and Hohsaka [8] have proposed a method using the protein-synthesis mechanism combined with four-base codon techniques to simulate a computation (accepting) process of finite automata. An important objective of this paper is to execute the proposed method on *E. coli* in order to improve the efficiency of the method and further develop a programmable *in vivo* computer. We describe the proposed method using an example of simple finite automaton, illustrated in Fig. 3, which is of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepts input strings with even numbers of symbol 1 and rejects input strings with odd numbers of 1s.

The input symbol '1' is encoded to the four-base subsequence `AGGU` and an input string is encoded into an mRNA by concatenating `AGGU` and `A` alternately and adding `AAUAAC` at the 3'-end. This one-nucleotide `A` in between `AGGU` is used to encode two states $\{s_0, s_1\}$, which is a same technique presented in [9]. For example, a string "111" is encoded into an mRNA:

$$\underbrace{\texttt{AGGU}}_{1}\texttt{A}\underbrace{\texttt{AGGU}}_{1}\texttt{A}\underbrace{\texttt{AGGU}}_{1}\texttt{AAAUAAC}.$$

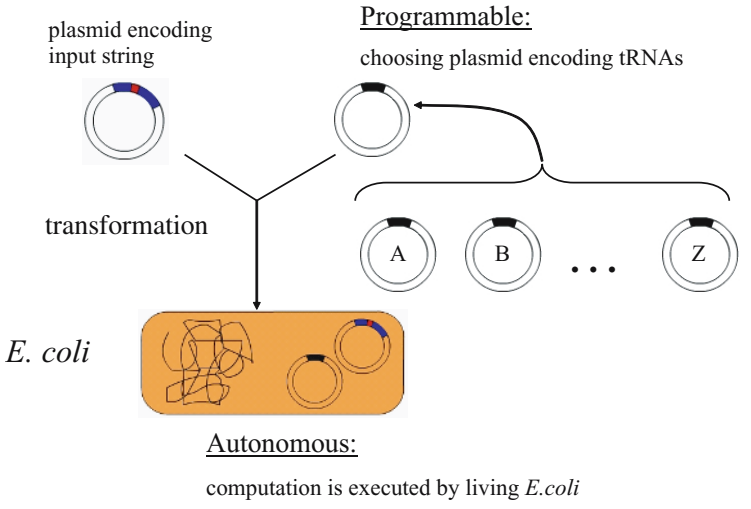

Fig. 2. A programmable and autonomous *in vivo* computer system based on *E. coli*

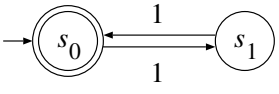

Fig. 3. A simple finite automaton of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepting input strings with even numbers of symbol 1 and rejecting input strings with odd numbers of 1s

(This encoding will be replaced with other four-base encoding in real laboratory experiments because of the translation efficiency.) The four-base anticodon (3')UCCA(5') of tRNA encodes the transition rule $s_0 \xrightarrow{1} s_1$, that is a transition from state s_0 to state s_1 with input symbol 1, and the combination of two three-base anticodons (3')UUC(5') and (3')CAU(5') encodes the rule $s_1 \xrightarrow{1} s_0$. Further, the encoding mRNA is linked to *lacZ*-coding RNA subsequence as a reporter gene for the detection of successful computations. Together with these encodings and tRNAs containing four-base anticodon (3')UCCA(5'), if a given mRNA encodes an input string with odd numbers of symbol 1 , an execution of the *in vivo* protein-synthesis system stops at the stop codon, which implies that the finite automaton does not accept the input string, and if a given mRNA encodes even numbers of 1s, the translation goes through the entire mRNA and the detection of acceptance is found by the *blue* signal of *lacZ*. Examples of accepting processes are shown in Fig. 4: (Upper) For an mRNA encoding a string "1111", the translation successfully goes through the entire mRNA and translates the reporter gene of *lacZ* which emits the blue signal. (Lower) For an mRNA encoding a string "111", the translation stops at the stop codon UAA, does not reach to the *lacZ* region and produces no blue signal.

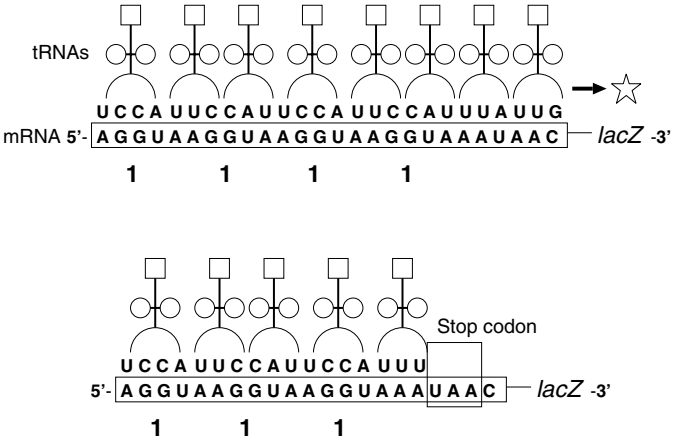

Fig. 4. Examples of accepting processes: (Upper) For an mRNA encoding a string "1111", the translation successfully goes through the mRNA and translates the reporter gene of *lacZ* emitting the blue signal. (Lower) For an mRNA encoding a string "111", the translation stops at the stop codon `UAG`, does not reach to the *lacZ* region and produces no blue signal.

If the competitive three-base anticodon (3')`UCC`(5') comes faster than the four-base anticodon (3')`UCCA`(5'), the incorrect translation (computation) immediately stops at the following stop codon `UAA`.

2.3 Designing Laboratory Protocols Using *E. coli*

In order to practically execute the laboratory experiments for our *in vivo* finite automata described in the previous sections, we have designed the following details of laboratory protocol. For the translation efficiency, we use tRNA with "`UCCU`" four-base anticodon in place of "`UCCA`".

(1) Construction of plasmid for tRNA with UCCU anticodon. The gene encoding a serine-inserting frameshift suppressor tRNA (designated as FS-Sup tRNA) [6] was generated by polymerase chain reaction (PCR) with four synthetic oligomers shown in Table 1. This PCR was performed using Pyrobest DNA polymerase (Takara Shuzo, Kyoto, Japan) and Gene Amp PCR System 2700 (ABI). The PCR product, after treated with MicroSpin Columns (QIAGEN), was digested with BamHI and Eco52I, and was then ligated into the corresponding sites of a derivative of pACYC184, by using Ligation kit ver.1 (Takara), to create plasmid pFSSuptRNA. This derivative of pACYC184 contains the lpp promoter before the BamHI site and the rrnC terminator after the Eco52I site. The use of these promoter and terminator for expressing tRNA in *E. coli* has been reported in [7]. *E. coli* MV1184 ElectroCells (Takara a) was transformed by electroporation with pFSSuptRNA and incubated in SOC medium at 37°C. The cells were then transferred onto LB Lennox plates (Nacarai) containing chloramphenicol (Wako) of 25 μg/ml to be inoculated at 37°C overnight. To extract

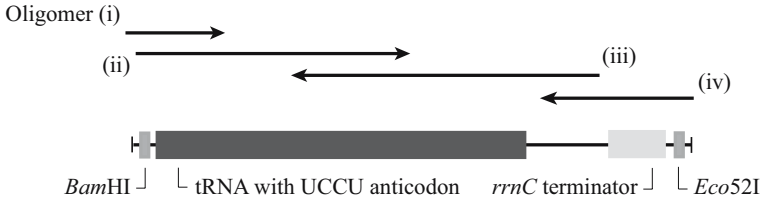

Fig. 5. Construction of FSSup tRNA with UCCU anticodon

Table 1. Oligomers for frameshift suppressor tRNA

Oligomer	Sequence
(i)	(5') CACACAGGATCCCCGTGGAGAGATGC (3')
(ii)	(5') GGATCCCCGTGGAGAGATGCCGGAGCGGCTGAACGGACCGGTCTTCCT AAACCGGAGTAGGGGCAAC (3')
(iii)	(5') GCTTTCGCTAAGGATCGTCGACTTTGGCGGAGAGAGGGGGATTTGAAC CCCCGGTAGAGTTGCCCCTACTCCGGTTTAG (3')
(iv)	(5') CACACACGGCCGTAAAAAAAATCCTTAGCTTTCGCTAAGGATCGTCG (3')

the plasmid, the cells from one colony were transplanted to LB Lennox medium of 1.5 ml containing chloramphenicol 25 μg/ml and cultured overnight at 37°C. Plasmid DNA from the cells was extracted by using QIAprep Spin Miniprep kit (Qiagen). Finally, the sequence of the FSSup tRNA gene was confirmed by sequencing using the standard dideoxy method.

(2) Plasmid carrying an encoded input string. DNA fragment carrying an encoded input string was made by annealing two oligomers, phosphorylated by T4 polynucleotide kinase (Toyobo), in an H buffer (Toyobo) with a thermal program of 95°C 2 mim followed by slow cooling to room temperature. The obtained fragment had overhanging bases at either end to be ligated into the PstI-XbaI sites pUC19 (Takara) (See Fig. 6). Amplification and sequence confirmation of this plasmid, pUC19 with the encoded input string, was performed as described in (1) except for a use of ampicillin (Nacarai) of 50 μg/ml in place of chloramphenicol.

(3) Cell preparation for electroporation. *E. coli* MV1184 with pFSSup-tRNA was cultured overnight in LB Lennox (3ml). This overnight culture was

Fig. 6. Encoded input string

Table 2. Oligomers for an encoded input string

Oligomer	Sequence
(v)	(5') GC$\underbrace{\text{AGGTA}\cdots\text{AGGTA}}_{\text{AGGTA}\times\text{n}}$AATAACACT (3')
(vi)	(5') CTAGAGTGTTATT$\underbrace{\text{TACCT}\cdots\text{TACCT}}_{\text{TACCT}\times\text{n}}$GCTGCA (3')

added to LB Lennox (150 ml) containing chloramphenicol 25 μg/ml, and inoculated at 37°C until OD_{595} becomes $0.6 \sim 0.8$. The fresh culture, thus prepared, was cooled on ice. Then, the culture was centrifuged at 5,000Xg for 15 min at 4°C, and then the supernatant was discarded and the pellet was re-suspended in cold water. This step of cell wash was repeated. The pellet thus obtained was suspended in 10 % glycerol, and was then centrifuged at 5,000Xg for 15 min at 4°C. After discarding the supernatant, the pellet was suspended in 10 % glycerol again. This cell suspension was applied to flash freezing with liquid nitrogen for store at -80°C.

E.coli MV1184 with pACYC184 instead of pFSSuptRNA was similarly treated for preparing cells for electroporation. MV1184 with pACYC184 was used for a control experiment.

(4) Calculation. The cells prepared in (3) were transformed with the plasmids carrying an encoded input string by electroporation. The transformed cells were added together with IPTG and X-Gal onto LB Lennox plates containing chloramphenicol of 25 μg/ml and ampicillin of 50 μg/ml. The cells were grown at 37°C overnight.

3 Experiments

We have done some laboratory experiments by following the laboratory protocols presented in Section 2.3 to execute the finite automaton shown in Fig. 3, which is of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepts input strings with even numbers of symbol 1 and rejects input strings with odd numbers of 1s.

We tested our method for six input strings, "1", "11", "111", "1111", "11111", and "111111", to see whether the method correctly accepts the input string "11", "1111", "111111", and rejects the strings "1", "111", "11111".

The results are shown in Fig. 7. Blue-colored colonies which indicates the expression of *lacZ* reporter gene have been observed only in the plates for the input strings 11, 1111, and 111111. Therefore, our *in vivo* finite automaton has succeeded to correctly compute the six input strings, that is, it correctly accepts the input strings 11, 1111, 111111 of even numbers of symbol '1' and correctly rejects 1, 111, 11111 of odd number of 1s. To our knowledge, this is the first experimental development of *in vivo* computer and has succeeded to execute an finite-state automaton on *E. coli*.

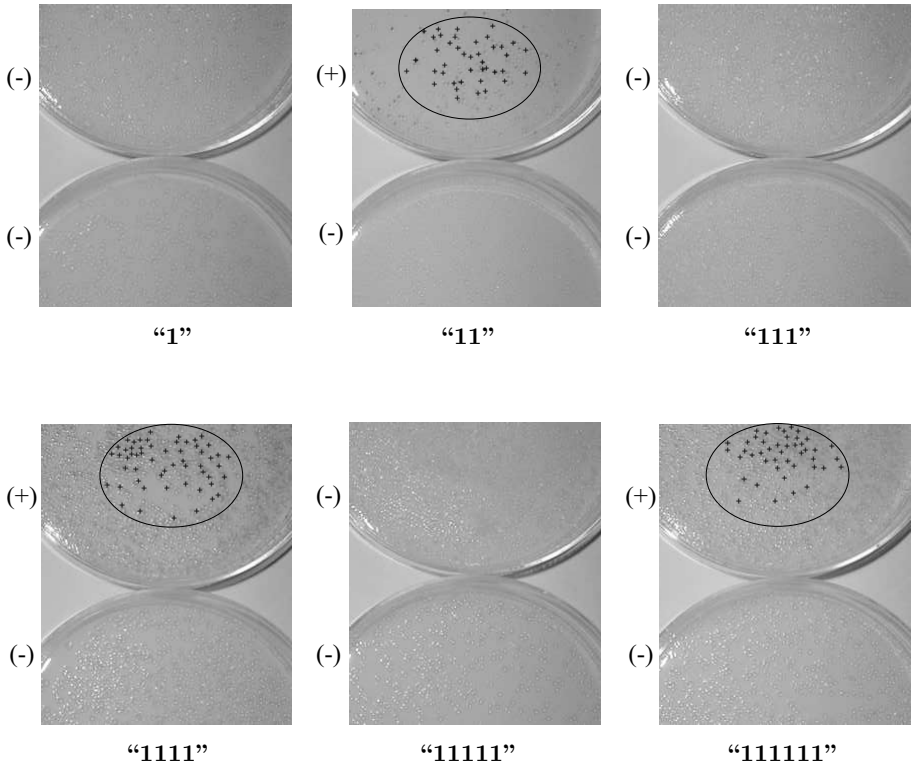

Fig. 7. Computation by the *E. coli* cells with plasmids of the input strings: 1, 11, 111, 1111, 11111, 111111. In each panel, the upper plate (part of a LB plate) shows the result in the presence of the suppressor tRNA with UCCU anticodon in the cell, while the lower plate shows the result of control experiment with no suppressor tRNA expressed. The signs (+) and (-) indicate the theoretical values about the expressions of *lacZ* reporter gene: (+) means that the cultured *E. coli* cells must express *lacZ* theoretically, and (-) means it must not express. Circles indicate the blue-colored colony expressing *lacZ*. Therefore, our *in vivo* finite automaton has correctly computed the six input strings, that is, it correctly accepts the input strings 11, 1111, 111111 of even numbers of symbol '1' and correctly rejects 1, 111, 11111 of odd number of 1s.

4 General Theory to Implement Finite Automata Using *n*-Base Codons

A general theory to implement any kinds of finite automata and any input strings on any alphabet is described as follows.

First, in theory, we assume that n-base codons (for arbitrary $n = 3, 4, 5, \ldots$), tRNAs containing the complementary n-base anticodons, and the *in vivo* protein-synthesis mechanism are available.

Next, we implement a finite automaton using n-base codons and some specific encodings. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a (deterministic) finite automaton, where

Q is a finite set of states numbered from 0 to k, Σ is an alphabet of input symbols, δ is a state-transition function such that $\delta : Q \times \Sigma \longrightarrow Q$, q_0 is the initial state, and F is a set of final states.

For the alphabet Σ, we encode each symbol a in Σ into a DNA subsequence, denoted $e(a)$, of fixed length. For an input string w on Σ, we encode $w = x_1 x_2 \cdots x_m$ into the following DNA subsequence, denoted $e(w)$:

$$e(x_1)\underbrace{\mathtt{AA}\ldots\mathtt{A}}_{k \text{ times}} e(x_2)\underbrace{\mathtt{AA}\ldots\mathtt{A}}_{k \text{ times}} \ldots e(x_m)\underbrace{\mathtt{AA}\ldots\mathtt{A}}_{k \text{ times}}$$

For the state-transition function from state q_i to state q_j with input symbol $a \in \Sigma$, we encode $\delta(q_i, a) = q_j$ into tRNA containing the following anticodon:

$$(3')\ \underbrace{\mathtt{UU}\ldots\mathtt{U}}_{i \text{ times}}\ c(e(a))\ \underbrace{\mathtt{UU}\ldots\mathtt{U}}_{k-j \text{ times}}\ (5')$$

where $c(y)$ denotes the complementary sequence of y. Thus, we represent each state in Q by the length of DNA sequence. This is the same technique presented in [9]. Finally, we add some specific DNA subsequence containing stop codons at the 3'-end of the encoding sequence $e(w)$. This is for the *in vivo* protein-synthesis system to stop a translation if the finite automaton does not accept an input string.

It would be easy to see that the protein-synthesis mechanism of *E. coli* with these specific encodings of the input string and tRNAs containing the anticodons encoding the state-transition function will simulate the computation process of a target finite automaton.

In practice, several four-base anticodons such as `AUCU`, `GGGA` and `GAUC` are executable [6] and some five-base anticodons [1] have been proved in laboratory experiments.

5 Discussions

The presented experiments of our *in vivo* finite automata based on *E. coli* propose a kind of population computations in the following two senses: (1) While a computation by one single *E. coli* cell is not effective and accurate, a colony consisting of a large number of *E. coli* cells provides a reliable computation. (2) Since one bacterial cell can multiply to over a million cells overnight, our *in vivo* computation framework offers a massively parallel computation. Further, our *in vivo* finite automata have a quite distinguished feature that an *in vivo* finite automaton is implemented in a living *E. coli* cell; it is not implemented simply by an incubation at a certain temperature.

Acknowledgements

This work was also performed in part through Special Coordination Funds for Promoting Science and Technology from the Ministry of Education, Culture,

Sports, Science and Technology, the Japanese Government, and a grant of Keio Leading-edge Laboratory of Science and Technology (KLL) specified research projects.

References

1. Anderson, J. C., T. J. Magliery, and P. G. Schultz. Exploring the limits of codon and anticodon size. *Chemistry & Biology*, 9, 237–244, 2002.
2. Benenson, Y., T. Paz-Ellzur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414, 430–434, 2001.
3. Bishop, R. E., B. K. Leskiw, R. S. Hodges, C. M. Kay, and J. H. Weiner. The entericidin locus of *Escherichia coli* and its implications for programmed bacterial cell death. *Journal of Molecular Biology*, 280, 583–596, 1998.
4. Hohsaka, T., Y. Ashizuka, H. Taira, H. Murakami, M. Sisido. Incorporation of nonnatural amino acids into proteins by using various four-base codons in an *Escherichia coli* in vitro translation system. *Biochemistry*, 40, 11060–11064, 2001.
5. Hohsaka, T., Y. Ashizuka, H. Murakami, M. Sisido. Five-base codons for incorporation of nonnatural amino acids into proteins. *Nucleic Acids Research*, 29, 3646–3651, 2001.
6. Magliery, T. J., J. C. Anderson, and P. G. Schultz. Expanding the genetic code: selection of efficient suppressors of four-base codons and identification of "shifty" four-base codons with a library approach in *Escherichia coli*. *Journal of Molecular Biology*, 307, 755–769, 2001.
7. Normanly, J., J. M. Masson, L. G. Kleina, J. Abelson, and J. H. Miller. Construction of two Escherichia coli amber suppressor genes. Proceeding of the National Academy of Sciences USA, 83, 6548–6552, 1986.
8. Sakakibara, Y. and T. Hohsaka. In Vitro Translation-based Computations. *Proceedings of 9th International Meeting on DNA Based Computers*, Madison, Wisconsin, 175–179, 2003.
9. Yokomori, T., Y. Sakakibara, and S. Kobayashi. A Magic Pot : Self-assembly computation revisited. *Formal and Natural Computing*, LNCS 2300, Springer-Verlag, 418–429, 2002.