Towards Molecular Programming

Masami Hagiya

JST CREST and Department of Computer Science,
Graduate School of Information Science and Technology, University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, JAPAN. hagiya@is.s.u-tokyo.ac.jp

Summary. In this paper, I first describe the directions of current research in the field of DNA and molecular computing by summarizing a recent international conference in this field: *DNA8, the Eighth International Meeting on DNA Based Computers*. Research in this field has clearly shifted from purely mathematical computations to broader applications in nanotechnology and biotechnology, and the principles and methods for designing molecular systems with information-processing capability for such applications are considered important. We call research into designing such molecular systems *molecular programming*. This paper reviews existing models of DNA and molecular computation and analyzes the results of these models, and then briefly describes some methods for molecular programming, including sequence design. It finally touches on molecular machines made of DNA, the current focus of molecular programming.

1 Introduction

Several years have passed since the direction of research in the field of DNA and molecular computing shifted from purely mathematical computation to much broader applications in nanotechnology and biotechnology [12, 13]. This means that principles and methods developed in the field are now being applied to non-mathematical problems, such as constructing molecular machines and analyzing the human genome. A few years ago, while developing his DNA computer for solving 3-SAT problems [47], Suyama insisted that if a DNA computer could solve 3-SAT problems accurately, then the computer could be used to analyze the human genome and produce reliable results that could be used for medical diagnosis [40, 35].

Another possible direction in research is towards applying evolutionary computation to molecular evolution. There was once interest in performing evolutionary computation with molecules, but now it is thought that applying methods established in evolutionary computation, such as the genetic algorithm or programming, to molecular evolution will be more fruitful [31]. Since

evolution can be regarded as a kind of computation, this direction is considered to be yet another application of molecular computing in a broad sense.

These new directions in the field were also topics at a recent international conference on DNA and molecular computing, i.e., *DNA8, the Eighth International Meeting on DNA Based Computers* [14], which was held in Sapporo, Japan, on June 10–13, 2002 (Fig. 1). Here, I briefly touch on the presentations at the conference and summarize current research trends.

Research in this field is increasingly being directed towards developing principles and methods for designing molecules and molecular systems that not only solve purely computational problems, but also the broader problems mentioned above. To solve such problems, a computing perspective is still important, because some kind of informationprocessing is always involved. For example, molecular machines should have information processing capability at the molecular level. Analyzing human genomes is nothing but processing information on genes and their products. Any molecular system that is capable of information processing is considered a subject of this field.

In order to construct such molecular systems, we need to have a deep understanding of what molecules can compute. This is exactly the question that researchers in the field of DNA and molecular computing have been hoping to answer, and many models of DNA and molecular computation have been proposed. In this paper, I very briefly summarize the models used for molecular computation and the results of analyses of the computational power of the models, including computability and complexity.

The increasing power of these models is allowing the field to expand as they are applied to broader applications. This requires research on design and construction. I use the term *molecular programming* to include research into designing molecules and molecular systems with information processing capabilities. The word *programming* suggests that designing molecules and molecular systems is like programming electronic computers. A typical example of molecular programming is designing DNA nucleotide sequences. Sequence design is one of the most important aspects of DNA computing, because it greatly influences the accuracy and efficiency of DNA computation. By hybridization, DNA molecules interact with each other and form complex structures or even machines encoded in their sequences. Moreover, the sequences of molecules with enzymatic functions, such as RNA and proteins, control their own reactions. Designing the sequences of such molecules is simply programming their behavior.

Needless to say, programming molecules and molecular systems involves many complex computational problems that are themselves good research subjects in computer science. In this paper, I briefly describe some methods of molecular programming, including sequence design. Finally, I touch on molecular machines made of DNA, which are currently the main target of molecular programming.

2 DNA8

DNA8, the Eighth International Meeting on DNA Based Computers, was held on June 10–13, 2002, at Hokkaido University, in Sapporo, Japan [14]. There were about 100 participants, including around 50 from overseas. I chaired the programming committee, and Azuma Ohuchi, of Hokkaido University, chaired the organizing committee. This conference series was organized under the aegis of the steering committee chaired by Grzegorz Rozenberg at the University of Leiden.

Fig. 1. DNA8

The sessions at DNA8 are listed below, along with some of the prominent speakers, some of whom were invited to speak.

- Molecular Evolution
  - Willen P. C. Stemmer (invited)
- Computing by Self-Assembly

  - Nadrian C. Seeman
- DNA Computing and Nanotechnology
  - Tomoji Kawai (invited)
  - John H. Reif and Thomas H. LaBean
- Applications to Graph Problems
- Applications to Biotechnology and Engineering
  - Akira Suyama (invited)
  - Ron Weiss
- Nucleic Acid Sequence Design
- Theory
  - Tom Head (invited)
  - Pierluigi Frisco and Sungchul Ji
- Autonomous Molecular Computation
  - Bernard Yurke (invited)
  - John H. Reif

The sessions *Computing by Self-assembly*, *DNA Computing and Nanotechnology*, and *Autonomous Molecular Computation* were all concerned with DNA nanotechnology, an emerging subfield of nanotechnology that is based on DNA and related molecules. In particular, the session *Autonomous Molecular Computation* was about molecules that move, i.e., molecular machines.

There were two particularly interesting talks in the session on *Applications to Biotechnology and Engineering*. One was the invited talk by Suyama about applications of DNA computing to genomic analysis. The other was by Basu, Karig and Weiss, who are trying to engineer signal processing in *E. coli* to make a sensor that detects a band at a particular molecular concentration [4].

The session on *Nucleic Acid Sequence Design* included many interesting talks about sequence design. In DNA8, interest moved towards secondary structure design, i.e., designing the sequences that fold into intended secondary structures [15].

In summary, classical Adleman-style DNA computing seems to be disappearing, while applications to nanotechnology and biotechnology are becoming central issues.

- The field of DNA nanotechnology treats both the construction and motion of structures.
- Even sequence design is geared towards DNA nanotechnology, as secondary structure design is becoming an important issue.
- Applications in genomic analysis are also becoming realistic. For example, Suyama proposed using his DNA computer for gene expression analysis and SNP analysis. Cell engineering is not a dream any more. Weiss *et al.* are trying to engineer cells to make sensors. This is a movement towards *in vivo* molecular computing [41].
- In the past, DNA computing flourished as a result of theoretical contributions related to the splicing system (or H-system) [28] and membrane system (or P-system) [27]. There were some interesting and important talks in

the theory session of DNA8. Nevertheless, theoretical contributions seem stalled, and the theory of DNA computing needs breakthroughs.

Lastly, I must mention that a number of chemists attended the conference, including Tomoji Kawai of Osaka University, who gave an invited talk about DNA nanotechnology, and Sen, who talked about electron transfer in DNA. Research on nanotechnology will involve increasing cooperation with chemistry and increasing numbers of chemists will likely develop an interest in the field.

3 Models for Molecular Computation and Their Analysis

In this section, I briefly summarize the models for molecular computation and analyze their computability and complexity. The analysis is intended to reveal the computational power of molecules and molecular reactions.

3.1 Computational Models

Adleman-Lipton Paradigm and its Refinements

I do not intend to explain the Adleman-Lipton paradigm in detail here [1, 22]. It is the first paradigm for DNA computing, and became the basis of many succeeding computational paradigms. It is a paradigm for solving a combinatorial search problem by 1) randomly generating solution candidates by hybridization of DNA, where each hybridized DNA molecule encodes a candidate, and 2) extracting solutions using data-parallel computation, where each experimental operation in a test tube is applied in parallel to all the molecules in the test tube.

I will mention two refinements of the paradigm. To solve 3-SAT problems, Suyama proposed a molecular algorithm in which solution candidates are not generated at once, but partial candidates are generated and gradually extended to form complete ones [47]. Once the partial candidates have been generated, those that can never be completed are immediately removed. This strategy reduces the number of candidates generated during computation. Suyama calls the strategy *dynamic programming*, because variables are ordered and each variable is processed according to the result of processing its preceding variables. Ogihara and Ray proposed a similar algorithm that they called *counting* [26].

Sakamoto *et al.* developed another refinement for 3-SAT problems [37]. Their machinery is called the *SAT Engine*, and it makes use of hairpin structures in DNA molecules (Fig. 2). In the SAT Engine, complementary literals are encoded by complementary nucleotide sequences in the sense of Watson and Crick. If a single-stranded DNA molecule contains two literals that are inconsistent with each other, i.e., a variable and its negation, then the

molecule forms a hairpin. This means that inconsistent assignments correspond to molecules containing a hairpin, so a SAT problem can be solved by removing hairpin molecules and checking whether consistent assignments remain.

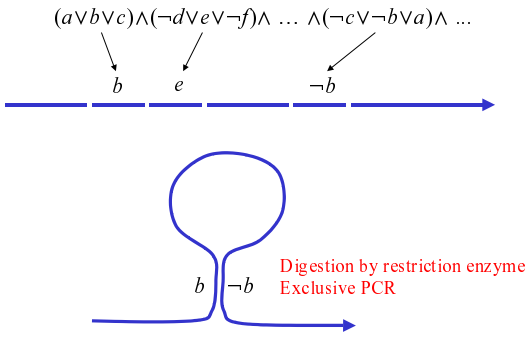

Fig. 2. Sakamoto's SAT Engine

Seeman-Winfree: Computing by Self-Assembly

Dr. Nadrian Seeman at the University of New York is a central figure in DNA nanotechnology. He received the 1995 Feynman Prize in Nanotechnology at the Fourth Foresight Conference on Molecular Nanotechnology for his research on the synthesis of 3-dimensional objects from DNA. Seeman has invented various DNA structures, while Winfree at the California Institute of Technology has proposed computational models based on self-assembly of those structures.

Among the structures that Seeman invented are *DNA tiles* [39]. *Double-crossover molecules* are examples of DNA tiles (Fig. 3, lower left). Each double-crossover molecule is made of two double-stranded molecules that exchange single strands at two points. Therefore, the tile is composed of four single-stranded DNA molecules that self-assemble.

Double-crossover molecules have four sticky ends, which each hybridize with the sticky end of another molecule (Fig. 3, right). Therefore, they can self-assemble and form a planar structure [42]. This process of self-assembly corresponds to allowing square tiles with colored edges to hybridize only if adjacent edges are of the same color (Fig. 3, upper left).

As mentioned below, the process of tiling square tiles with colored edges is universal [43], because it can simulate execution of one-dimensional cellular automata.

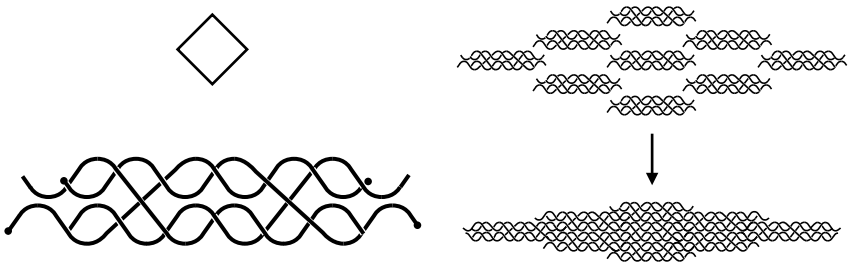

Fig. 3. Seeman's DNA tiles.

Head: Splicing System

The *splicing system* introduced by Head, also known as the *H-system*, is a formal model of DNA recombination, i.e., digestion by restriction enzymes and concatenation by ligase [28]. It is a model that can be used to generate languages, by regarding nucleotide sequences as strings of the letters $\{\mathrm{A}, \mathrm{G}, \mathrm{C}, \mathrm{T}\}$.

Digestion and ligation are modeled using a set of splicing rules. A *splicing rule* is a string of the form $u1\$u2\#u3\$u4$. For the splicing rule $r = u1\$u2\#u3\$u4$, we define the relation $\vdash_r$ between pairs of strings as follows:

$$(x_1u_1u_2x_2, y_1u_3u_4y_2) \vdash_r (x_1u_1u_4y_2, y_1u_3u_2x_2)$$

Let R be a set of splicing rules, and A a set of initial strings called axioms. Then, the language L generated by R and A is inductively defined as follows:

- If $x \in A$, then $x \in L$.
- If $x, y \in L$, $r \in R$ and $(x, y) \vdash_r (z, w)$, then $z, w \in L$.

It is known that if R and A are finite, then L is regular.

Other Models

Other models for molecular computation include the *Sticker Model* [33], which is another enhancement of the Adleman-Lipton Paradigm. There are also models for parallel computation of Boolean circuits, such as Ogihara and Ray's model [25]. Many models have been proposed for molecular state machines or finite automata, including those of Hagiya and Sakamoto [11, 34, 19] and Shapiro [6]. The last two will also be referred to at the end of this paper.

3.2 Computability

Considering the computability of molecular computations, I mention two typical results here: one by Winfree concerning the computability of self-assembly

of DNA, and the other by Head and his colleagues, who extended the splicing model in various ways to achieve universality of gene recombination.

The computational power of self-assembly by DNA molecules has been thoroughly investigated by Winfree [43]. He realized that languages generated by the hybridization of linear molecules are regular. Here, by linear molecules, I mean double-stranded DNA molecules with sticky ends that allow them to hybridize with each other. The set of all double-stranded molecules that can be generated by hybridization of copies of given linear molecules is regular, if we consider double-stranded molecules as strings of the letters $\{\mathrm{A}, \mathrm{G}, \mathrm{C}, \mathrm{T}\}$. Winfree pointed out that languages generated by linear, hairpin, and 3-junction molecules are context-free. By hybridizing copies of those molecules with their sticky ends and ligating them together, we obtain single-stranded DNA molecules after denaturation. The set of these single strands is context-free as a language over $\{\mathrm{A}, \mathrm{G}, \mathrm{C}, \mathrm{T}\}$.

Finally, Winfree showed that languages generated by linear and double-crossover molecules are recursively enumerable. This beautiful result was obtained by simulating a one-dimensional cellular automaton using the two-dimensional self-assembly of DNA tiles. Note that one-dimensional cellular automata are Turing complete because they can simulate Turing machines.

Another important result concerning the computability of molecular computations was obtained by various extensions to the splicing model. As mentioned before, the original splicing system can only generate regular languages. Therefore, many theoreticians explored ways to extend the model to achieve universal computability [28]. Some extensions that are universal are:

- splicing circular molecules,
- using multiple tubes, and
- making rules time-varying.

3.3 Complexity

The computational complexity of molecular computation is measured in time and space as are other kinds of computation. In molecular computing, time complexity has two aspects: the number of laboratory operations and the time taken for each operation. In particular, the latter is important from the viewpoint of analyzing the computational power of molecular reactions. The space complexity of molecular computation corresponds to the number of required molecules and implies the degree of parallelism. The size or length of molecules should also be considered in order to accurately measure space complexity. In general, molecular computation gains time efficiency by sacrificing space efficiency, which is typical of the Adleman-Lipton Paradigm. Therefore, it is important to analyze the trade-off between the two measures.

Some classical examples of analyzing the complexity of molecular computation, in which time is measured only by the number of laboratory operations, include that of Reif, who showed that a nondeterministic Turing machine computation with input size n, space s, and time $2^{O(s)}$ can be executed in his PAM

Model using $O(s)$ PA-Match steps and $O(s \log s)$ other PAM steps, employing aggregates of length $O(s)$ [29]. Beaver showed that polynomial-step molecular computers compute PSPACE [5]. Roosand Wagner proved that the problems in $\mathrm{P}^{\mathrm{NP}} = \Delta_2^{\mathrm{P}}$ can be solved in polynomial time using Lipton's model [30].

As for the complexity of self-assembly, Rothemund and Winfree proved that for any non-decreasing unbounded computable function $f(N)$, the number of tiles required for the self-assembly of an $N \times N$ square is bounded infinitely often by $f(N)$ [32]. Winfree *et al.* showed that the linear assembly of string tiles can generate the output languages of finite-visit Turing Machines [45].

No chemical reaction can be executed instantaneously. The efficiency of a chemical reaction depends on the yield of the reaction, which is related to its equilibrium constant, K, and the time to reach the equilibrium, which is related to the reaction constant k. For example, in the reaction $A \leftrightarrow B$, the concentration of B, denoted by $[B]$, is given as a function of elapsed time t as follows.

$$[B] = (K/(1+K))(1 - e^{-(k+k_{-1})t})$$

In the equation, k and k_{-1} are the forward and backward reaction constants, respectively, and the equilibrium constant K is given by $K = k/k_{-1}$.

Note that no chemical reaction can escape error. A typical error in reactions involving DNA is mishybridization, which means that non-complementary nucleotide sequences hybridize. Since the probability of such error can never be zero, probabilistic analysis is essential in molecular computation. Some examples of probabilistic analysis of molecular computation follow. Karp *et al.* proved that the number of extract operations required to achieve error-resilient bit evaluation is $\Theta(\lceil \log_\epsilon \delta \rceil \times \lceil \log_\gamma \delta \rceil)$ [17]. Kurtz made a thermodynamic analysis of path formation in Adleman's experiment and showed that the time needed to form a Hamiltonian path is $\Omega(n^2)$ [20]. Winfree also made a thermodynamic analysis of DNA tiling in his Ph.D. thesis [44].

4 Molecular Programming

Molecular programming aims at establishing systematic design principles for molecules and molecular systems with information processing capabilities, and developing methods to ease their design and construction. In order to achieve this goal, we must solve many computational problems, which are also deep from the viewpoint of computer science.

Methods in molecular programming are roughly classified into those for designing molecules and those for designing molecular reactions. To design molecules, the problem of designing nucleotide sequences is an important issue in DNA computing. To design molecular reactions, models for DNA reactions, such as hybridization and denaturation, have been developed, as have methods for simulating those models using electronic computers. They are aimed at tuning reaction conditions to realize accurate and efficient reactions.

In this section, I briefly review the methods for sequence design used in DNA computing and those for simulating DNA reactions. I finally touch on molecular machines made of DNA, which are currently the main targets of molecular programming. They are molecules that change their state according to reactions with other molecules or environments.

4.1 Designing Molecules

Sequence Design

Sequence design has always been an important issue in DNA computing. It first aimed at designing a large library of nucleotide sequences that only hybridize with their complementary sequences, avoiding mishybridization. Research on sequence design began with defining a measure for evaluating the possibility of mishybridization in a library of nucleotide sequences. One well-established measure is the H-measure developed by Garzon *et al.*, which is computed using the Hamming distance, i.e., the number of mismatches, while considering frame shifts [10].

The genetic algorithm and related methods have been used to find libraries of nucleotide sequences with a maximum H-measure greater than a given threshold. Another solution to the problem was recently proposed by Arita [3] and elaborated on by Kobayashi [18]. Called the *template method*, it is an enhancement of the method originally proposed by Condon *et al.* [9]

A *template* is a sequence of zeros and ones. Either G or C is substituted for 1, and either A or T is substituted for 0. For example, from the template 011010, sequences such as $ACCTGA$, $TGCTCA$, and $TCGACA$ are obtained. Given a template of length n, a library of nucleotide sequences of size 2^n is obtained. Note that such a library satisfies the condition that all the sequences have the same GC content; therefore, their melting temperature is uniform.

In order to obtain a good library, template T should satisfy the condition that the following patterns always contain at least d mismatches with T with any frame shift.

$$T^R \quad TT^R \quad T^RT \quad TT \quad T^RT^R$$

In the pattern, T^R denotes the reverse of T. If $T = 110100$, then $T^R = 001011$. Using such a template, at least d mismatches are induced with shifting and reversal of sequences.

Here are some examples of templates. The following template has a length 6 and 2 mismatches:

110100

This is one of 2^6 such templates. For a length of 11 with 4 mismatches:

01110100100
01011100010
11000100101

For length 23 with 9 mismatches:

01111010110011001010000
10110011001010000011110
11100000101001100110101

Using a template of length n and with d mismatches, a library of DNA sequences is obtained by adopting an error-correcting code, such as BCH, Golay, Hamming, etc. If an n-bit code whose Hamming distance is d is used, then a library of DNA sequences is obtained by substituting G (or C) for 1 if the corresponding bit in a code word is 0 (or 1), and A or T for 0 if it is 0 (or 1). Given the conditions for the template, d mismatches are guaranteed with shifts and reverses. Otherwise, d mismatches are guaranteed with the code used.

Currently, sequence design that avoids both mishybridization with respect to the H-measure and unwanted secondary structures is being investigated [2].

Inverse Folding

Another direction in sequence design is designing a sequence that folds into a given secondary structure. This problem is called *inverse folding*, because it is the inverse of the problem of finding the secondary structure of a sequence with the minimum free energy. The inverse folding problem is to find a sequence whose minimum energy structure coincides with the given one.

The Vienna group has been working on the inverse folding problem and the folding problem [16], using dynamic programming as in the *mfold* of Zuker [49]. More accurately, the problem is formulated as follows. Let Ω be the target structure, and x be a sequence whose structure has a free energy of $E(x, \Omega)$. If $G(x)$ is the ensemble free energy of x, obtained using McCaskill's algorithm [24], then the probability p of x having the structure Ω is as follows:

$$p \;=\; \frac{e^{-E(x,\Omega)/RT}}{e^{-G(x)/RT}}$$

Therefore, the cost function $\Xi(x)$ of sequence x can be defined as:

$$\Xi(x) \;=\; E(x, \Omega) - G(x) \;=\; -RT \ln p$$

The problem is then reduced to finding x, such that $\Xi(x)$ is minimized. The free energy of a secondary structure is computed using the nearest neighbor model, a standard model based on stacking energies [7].

4.2 Designing Molecular Reactions

In order make molecular reactions accurate and efficient, it is important to tune the reaction conditions such as temperature, salt concentration, and time.

The order of operations is also important. Simulating molecular reactions is considered a useful tool to find appropriate reaction conditions.

Molecular reactions should be simulated using good models. For example, for DNA hybridization, the nearest neighbor model based on stacking energies is widely used [7, 36]. This model is also the basis for designing sequences that form target secondary structures. The staggering zipper model is an enhancement, in which hybridization consists of two steps. In the nucleation step, a nucleus of hybridization is formed between two strands, and in the zipping step, the nucleus is extended to form a complete hybrid.

Using such models, tools have been developed for simulating reactions among DNA molecules, such as e-PCR [38] and *the Virtual Nucleic Acid simulator* (VNA) [23].

VNA simulates reactions among molecules consisting of virtual bases [23]. A molecule in this simulator is a hybrid of virtual strands, such as:

```
abcd
  ||
  CDEF
```

Each character is a virtual base, representing a nucleotide sequence of appropriate length, say 10 bases. Corresponding upper- and lowercase letters represent complementary sequences. The simulator supports the following reactions among such molecules.

- reactions
- hybridization
- denaturation
- digestion
- ligation
- self-hybridization
- extension

The simulator can verify the feasibility of algorithms for DNA computing, verify the validity of molecular biology experiments, such as PCR, and fit parameters in molecular biology experiments. Examples include Ogihara and Ray's computation of Boolean circuits [25], Winfree's construction of double-crossover molecules, and simulated PCR experiments.

VNA is implemented in Java and is executable as an applet. In order to simulate reactions among dynamically generated molecules, it combines combinatorial enumeration of molecules with continuous simulation based on differential equations. Interestingly, the concentrations of molecules are computed using differential equations to avoid a combinatorial explosion by setting a threshold on the concentration of a newly generated molecule. This also supports stochastic simulation.

Although it is still far from being able to simulate real reactions faithfully, because most of the reaction constants are not known, VNA has been used

to examine the possibility of combining various reactions to find expected or unexpected reaction paths.

In general, it is becoming increasingly important to grasp both the static and dynamic behavior of molecules, because research in DNA and molecular computing is moving toward designing molecules that move or change their states.

One of the next steps in this direction will be to estimate the energy barrier between two secondary structures of DNA or RNA. The Vienna group has already estimated energy barriers to design multi-state RNA molecules, and this is considered an important area of future research, particularly for designing molecular machines [8].

4.3 Molecular Machines

As mentioned in the previous section, there are many proposals and experiments for designing and implementing molecular machines out of DNA. Some of these include:

- Seeman's DNA motor using B-Z transition [21]
- Hagiya's whiplash PCR [11, 34, 19] (Fig. 4)
- Yurke's molecular tweezers [48]
- Seeman's PX-JX2 switch [46]
- Shapiro's DNA automata [6] (Fig. 5)

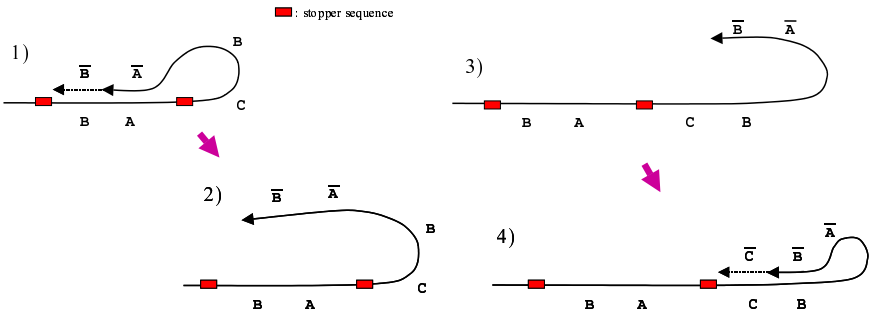

Fig. 4. Hagiya's Whiplash PCR

5 Concluding Remarks

In this paper, I summarized the research in DNA and molecular computing and research trends observed at DNA8, the recent international conference in

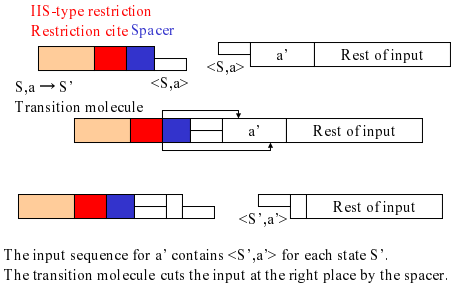

Fig. 5. Shapiro's DNA Automata

this field, to explain the importance of research on molecular programming, i.e., designing molecules and molecular systems with information-processing capabilities.

In Japan, a research group on molecular programming funded as *Priority Area Research* by the Ministry of Education has recently been formed. In addition, a CREST project has been funded by the Japan Science and Technology Corporation to research *molecular addressing* for constructing molecular memory out of DNA and related molecules. It is also intended to explore new directions in molecular computing, including *optical molecular computing.*

The preparation of this paper was supported by both of these projects.

References

1. Adleman, L.M.: Molecular computation of solutions to combinatorial problems, Science **266**, 1021–1024 (1994)
2. Andronescu, M., Dees, D., Slaybaugh, L., Zhao, Y., Condon, A.E., Cohen, B., Skiena, S.: Algorithms for testing that DNA word designs avoid unwanted secondary structure. In: *Preliminary Proceedings of the Eighth International Meeting on DNA Based Computers* (2002) pp 92–104
3. Arita, M., Kobayashi, S.: DNA sequence design using templates, New Generation Computing **20**, 263–277 (2002)
4. Basu, S., Karig, D., Weiss, R.: Engineering signal processing in cells: Towards molecular concentration band detection. In: *Preliminary Proceedings of the Eighth International Meeting on DNA Based Computers* (2002) pp 80–89
5. Beaver, D.: A universal molecular computer. In: *DNA Based Computers,* DIMACS Series in Discrete Mathematics and Theoretical Computer Science **27**, 29–36 (1996)
6. Benenson, Y., Paz-Elizur, T., Adar, R., Heina, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules, Nature **414**, 430–434 (2001)
7. Cantor, C.R., Schimmel, P.R.: *Biophysical Chemistry, Part III: The behavior of biological macromolecules* (W.H. Freeman and Company 1980)

8. Flamm, C., Hofacker, I.L., Maurer-Stroh, S., Stadler, P.F., Zehl, M.: Design of multistable RNA molecules, RNA **7**, 254–265 (2001)
9. Frutos A.G., Liu, Q., Thiel, A.J., Sanner, A.M.W., Condon, A.E., Smith, L.M., Corn, R.M.: Demonstration of a word design strategy for DNA computing on surfaces, Nucleic Acids Research **25**(23), 4748–4757 (1997)
10. Garzon, M., Neathery, P., Deaton, R.J., Murphy, R.C., Franceschetti, D.R., Stevens, S.E.Jr.: A new metric for DNA computing. In: *Proceedings of 2nd Annual Genetic Programming Conference* (1997) pp 472–478
11. Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., Yokoyama, S.: Towards parallel evaluation and learning of Boolean μ-formulas with molecules. In: *DNA Based Computers III.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science vol 48 (1999) pp 57–72
12. Hagiya, M.: Perspectives on molecular computing, New Generation Computing **17**(2), 131–140 (1999)
13. Hagiya, M.: From molecular computing to molecular programming. In *DNA Computing, 6th International Workshop on DNA Based Computers, DNA 2000.* Lecture Notes in Computer Science, vol 2054 (Springer, Berlin Heidelberg New York 2001) pp 89–102
14. Hagiya, M., Ohuchi, A. (eds.): *Preliminary Proceedings of the Eighth International Meeting on DNA Based Computers.* Hokkaido University (2002). http://hagi.is.s.u-tokyo.ac.jp/dna8/
Also to appear as Lecture Notes in Computer Science, vol 2568 (Springer, Berlin Heidelberg New York 2003)
15. Heitsch, C.E., Condon, A.E., Hoos, H.H.: From RNA secondary structure to coding theory: A combinatorial approach. In: *Preliminary Proceedings of the Eighth International Meeting on DNA Based Computers* (2002) pp 125–136
16. Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, L.S., Tacker, M., Schuster, P.: Fast folding and comparison of RNA secondary structures, Monatsh. Chem. **125**, 167–188 (1994)
17. Karp, R., Kenyon C., Waarts, O.: Error-resilient DNA computations. In: *Seventh ACM-SIAM Symposium on Discrete Algorithms* (1996) pp 458–467
18. Kobayashi, S., Kondo, T., Arita, M.: On template method for DNA sequence design. In: *Preliminary Proceedings of the Eighth International Meeting on DNA Based Computers* (2002) pp 115–124
19. Komiya, K., Sakamoto, K., Gouzu, H., Yokoyama, S., Arita, M., Nishikawa, A., Hagiya, M.: Successive state transitions with I/O interface by molecules. In: *DNA Computing, 6th International Meeting on DNA Based Computers, DNA 2000.* Lecture Notes in Computer Science, vol 2054 ((Springer, Berlin Heidelberg New York 2001) pp 17–26
20. Kurtz, S.A., Mahaney, S.R., Royer, J.S., Simon, J.: Active transport in biological computing. In: *DNA Based Computers II.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 44 (1999) pp 171–179
21. Mao, C., Sun, W., Shen, Z., Seeman ,N.C.: A nanomechanical device based on the B-Z transition of DNA, Nature **397**, 144–146 (1999)
22. Lipton, R.J.: DNA Solution of hard computational problems, Science **268**, 542–545 (1995)
23. Nishikawa, A., Yamamura, M., Hagiya, M.: DNA computation simulator based on abstract bases, Soft Computing **5**(1), 25–38 (2001)
24. McCaskill, J.S.: The equilibrium partition function and base pair binding probabilities for RNA secondary structure, Biopolymers **29**, 1105–1119 (1990)

25. Ogihara, M., Ray, A.: DNA based self-propagating algorithm for solving bounded-fan-in Boolean circuits. In: *Genetic Programming 98* (1998) pp 725–730
26. Ogihara, M., Ray, A.: DNA Based parallel computation by "counting". In: *DNA Based Computers III.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 48 (1999) pp 255–264
27. Păun, G.: *Membrane Computing — An Introduction* (Springer, Berlin Heidelberg New York 2002)
28. Păun, G., Rozenberg, G., Salomaa, A.: *DNA Computing* (Springer, Berlin Heidelberg New York 1998)
29. Reif, J.H.: Parallel molecular computation. In: *Seventh Annual ACM Symposium on Parallel Algorithms and Architectures* (1995) pp 213–223
30. Rooß, D., Wagner, K.W.: On the power of DNA-computing, Information and Computation **131**, 95–109 (1996)
31. Rose, J.A., Hagiya, M., Deaton, R.J., Suyama, A.: A DNA Based *in vitro* genetic program, Journal of Biological Physics **28**, *in press* (2002)
32. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing* (2000) pp 459–468
33. Roweis, S., Winfree, E., Burgoyne, R, Chelyapov, N.V., Goodman, M.F., Rothemund, P.W.K., Adleman, L.M.: A sticker based model for DNA computation. In: *DNA Based Computers II.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 44 (1999) pp 1–29
34. Sakamoto, K., Kiga, D., Komiya, K., Gouzu, H., Yokoyama, S., Ikeda, S., Sugiyama, H., Hagiya, M.: State transitions by molecules, Biosystems **52**, 81–91 (1999)
35. Sakakibara, Y., Suyama, A.: Intelligent DNA Chips: Logical operation of gene expression profiles on DNA computers. In: *Genome Informatics 2000.* Genome Informatics Series, vol 11 (Universal Academy Press, Inc., Tokyo, Japan 2000) pp 33–42
36. SantaLucia, J.Jr., Allawi, H.T., Seneviratne, P.A.: Improved nearest-neighbor parameters for predicting DNA duplex stability, Biochemistry **35**(11), 3555–3562 (1996)
37. Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M.: Molecular computation by DNA hairpin formation, Science **288**, 1223–1226 (2000)
38. Schuler, G.D.: Electronic PCR: bridging the gap between genome mapping and genome sequencing, Trends in Biotechnology **16**(11), 456–459 (1998) http://www.ncbi.nlm.nih.gov/genome/sts/epcr.cgi
39. Seeman, N.C., et al: The perils of polynucleotides: The experimental gap between the design and assembly of unusual DNA structures. In: *DNA Based Computers II.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 44 (1999) pp 215–233
40. Suyama, A., Nishida, N., Kurata, K., Omagari, K.: Gene expression analysis by DNA computing. In: *Currents in Computational Molecular Biology* (ISBN 4-946443-61-4, 2000) pp 12–13
41. Weiss, R., Knight, T.F.Jr.: Engineered communications for microbial robotics. In: *DNA Computing, 6th International Meeting on DNA Based Computers, DNA 2000.* Lecture Notes in Computer Science, vol 2054 (Springer, Berlin Heidelberg New York 2001) pp 1–16

42. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals, Nature **394**, 539–544 (1998)
43. Winfree, E., Yang, X., Seeman, N.C.: Universal computation via self-assembly of DNA: some theory and experiments. In: *DNA Based Computers II.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 44 (1999) pp 191–213
44. Winfree, E.: Simulations of computing by self-assembly. In: *Preliminary Proceedings, Fourth International Meeting on DNA Based Computers,* June 15 – June 19, 1998 (University of Pennsylvania 1998) pp 213–239
Also in Erik Winfree's PhD Thesis: *Algorithmic Self-Assembly of DNA* (California Institute of Technology May 19 1998)
45. Winfree, E., Eng, T., Rozenberg, G.: String tile models for DNA computing by self-assembly. In: *DNA Computing, 6th International Meeting on DNA Based Computers, DNA 2000.* Lecture Notes in Computer Science, vol 2054 (Springer, Berlin Heidelberg New York 2001) pp 63–88
46. Yan, H., Zhang, X., Shen, Z., Seeman, N.C.: A robust DNA mechanical device controlled by hybridization topology, Nature **415**, 62–65 (2002)
47. Yoshida, H., Suyama, A.: Solutions to 3-SAT by breadth first search. In: *DNA Based Computers V,* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 54 (1999) pp 9–22
48. Yurke, B., Turberfield, A.J., Mills, A.P.Jr., Simmel, F.C., Neumann, J.L.: A DNA-fuelled molecular machine made of DNA, Nature **406**, 605–608 (2000)
49. Zuker, M., Steigler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, Nucleic Acids Research **9**, 133–148 (1981)