Junghuei Chen
John Reif (Eds.)

LNCS 2943

DNA Computing

9th International Workshop on DNA Based Computers, DNA9 Madison, WI, USA, June 2003, Revised Papers

Springer

Lecture Notes in Computer Science 2943

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Junghuei Chen John Reif (Eds.)

DNA Computing

9th International Workshop
on DNA Based Computers, DNA9
Madison, WI, USA, June 1-3, 2003
Revised Papers

Springer

eBook ISBN: 3-540-24628-2
Print ISBN: 3-540-20930-1

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.kluweronline.com
and the Springer Global Website Online at: http://www.springeronline.com

Preface

Biomolecular computing is an interdisciplinary field that draws together molecular biology, DNA nanotechnology, chemistry, physics, computer science and mathematics. The annual international meeting on DNA-based computation has been an exciting forum where scientists of different backgrounds who share a common interest in biomolecular computing can meet and discuss their latest results. The central goal of this conference is to bring together experimentalists and theoreticians whose insights can calibrate each others' approaches. The 9th Annual International Meeting on DNA Based Computers was held during June 1–4, 2003 in the University of Wisconsin, Madison, USA. The meeting had 106 registered participants from 12 countries around the world.

On the first day of the meeting, we had three tutorials: the first was on self-assembly of DNA nano structures which focused on the basic techniques of using designed DNA nano molecules to be self-assembled onto larger structures for computational purposes. This tutorial was given by Hao Yan of Duke University. The second tutorial was given by Chengde Mao of Purdue University in which Dr. Mao presented basic DNA biochemistry that was designed for non experimentalists. The third tutorial was given by Max Garzon of the University of Memphis. Dr. Garzon gave a lecture on computational complexity which was tailored for non-computer scientists. The next three days were for invited plenary lectures, and regular oral and poster presentations. Invited plenary lectures were given by Helen Berman of Rutgers University (USA), Giancarlo Mauri of the University of Milan (Italy), Guenter von Kiedrowski of Ruhr University (Germany), and Sorin Istrail of Celera/Applied Biosystems.

The organizers sought to attract the most significant recent research with the highest impact on the development of the discipline. Papers and posters with new experimental results were particularly encouraged. Authors who wished their work to be considered for either oral or poster presentation were asked to select from one of two submission "tracks": Track A, Full Paper; Track B, One-Page Abstract.

For authors with late-breaking results, or who were submitting their manuscript to a scientific journal, a one-page abstract, rather than a full paper, could be submitted in Track B. Authors could (optionally) include a preprint of their full paper for consideration by the program committee. The program committee received 48 submissions in Track A, and 12 submissions in Track B. These submissions were then reviewed by the program committee members. In principle, four committee members were allocated for each submission. In considering the returned review reports, all discussions pertaining to the final decisions were made online by the program committee members. We finally selected 32 oral presentations from Tracks A and B. The oral and poster presentations included all areas that relate to biomolecular computing, such as algorithms and applications, analysis of laboratory techniques/theoretical models, computational

processes in vitro and in vivo, DNA-computing-based biotechnological applications, DNA devices, error evaluation and correction, in vitro evolution, models of biomolecular computing, molecular design, and simulation tools.

The editors would like to acknowledge the help of the conference's Program Committee in reviewing the submitted abstracts. The editors thank the Organizing Committee for their superb organization skill. We are grateful for the generous support and sponsorship of the conference by GenTel Corporation, DARPA (IPTO Biocomputation), NSF (CISE QUBIC ITR), and the Chemistry Department of the University of Wisconsin, Madison. Finally, the editors would like to thank all of the participants in the DNA9 conference for making it a wonderful experience. We hope that this volume has captured the spirit and exhilaration that we experienced at the conference.

December 2003

Junghuei Chen
John Reif

Organization

Program Committee

Martyn Amos	University of Exeter, UK
Junghuei Chen (Chair)	University of Delaware, USA
Russell Deaton	University of Arkansas, USA
Masami Hagiya	University of Tokyo, Japan
Natasha Jonoska	University of South Florida, USA
Lila Kari	University of Western Ontario, Canada
Laura Landweber	Princeton University, USA
Gheorghe Paun	Institute of Mathematics of the Romanian Academy, Romania
John Reif (Co-chair)	Duke University, USA
Ned Seeman	New York University, USA
Ehud Shapiro	Weizmann Institute of Science, Israel
Akira Suyama	University of Tokyo, Japan
Erik Winfree	California Institute of Technology, USA
Bernard Yurke	Bell Laboratories, Lucent Technologies, USA

Organizing Committee

Bryce Nelson	GenTel Corporation, Madison, Wisconsin, USA
Robert M. Corn	University of Wisconsin, Madison, Wisconsin, USA
Christine E. Heitsch	University of Wisconsin, Madison, Wisconsin, USA
Roberta M. Ostrander	University of Wisconsin, Madison, Wisconsin, USA
Lloyd M. Smith	University of Wisconsin, Madison, Wisconsin, USA

Sponsors

GenTel Corporation
DARPA (IIPTO Biocomputation)
NSF (CISE QUBIC ITR)
UW-Madison, Chemistry

This page intentionally left blank

Table of Contents

Self-Assembly and Autonomous Molecular Computation

Experimental Solutions

New Computing Models

A Lab-on-a-Chip Module for Bead Separation in DNA-Based Concept Learning

Hee-Woong Lim[1], Hae-Man Jang[2], Sung-Mo Ha[2], Young-Gyu Chai[2], Suk-In Yoo[1], and Byoung-Tak Zhang[1]

[1]Biointelligence Laboratory, School of Computer Science and Engineering
Seoul National University, Seoul 151-742, Korea
{hwlim,siyoo,btzhang}@bi.snu.ac.kr
[2]Department of Biochemistry and Molecular Biology,
Han-Yang University, Ansan, Kyongki-do 425-791, Korea
{hmjang,smha,ygchai}@bi.snu.ac.kr

Abstract. Affinity separation with magnetic beads is an important and widely used technique for DNA computing. We have designed and implemented an experimental lab-on-a-chip module for affinity-bead separation for DNA-based concept learning. Magnetic beads with DNA-probe sequences immobilized on their surface were used to select target strands, and these beads are restrained in the channel by a permanent magnet on top of the module. The separation process consists of two steps, i.e. hybridization and denaturation. We confirmed the separation process by a mixed solution that contains FITC modified strands, and measured the yield by UV spectrophotometer. The experimental results demonstrate a successful separation of the mixed DNA.

1 Introduction

Although one of the major attractions of DNA computing is the massive parallelism, the DNA computing operations involve a number of manual steps which require a large amount of time for bio-chemical reactions. This is one of the reasons that the application of DNA computing has been limited to the small-scale problems. Lab-on-a-chip technology provides a solution to this restriction. The miniaturization technology allows for integration and automation of experimental steps. It also reduces the amount of material necessary for the reaction and processing of the DNA. In addition, we are able to diminish the factors of errors inherent in DNA computing. Some examples for this technology can be found in [4,5,9,11,12].

Among many bio-lab techniques used in DNA computing, the affinity bead separation is an essential method for selecting the specific DNA strands. It served as a query method in a DNA-based database by associative search [2]. It was also used to check whether a path contains a specific city in HPP or TSP problems [1]. Moreover, Boolean value verification in the SAT problem [3] and attribute-value checking for hypothesis space refinement in the DNA-based concept learning [8] are accomplished by this technique.

However, in real experiment, the yield of this affinity separation is very low [6], and the information is lost or reduced while the operation is repeated. Thus, an amplification process such as PCR is indispensable to recover the information, which

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 1-9, 2004.

is the amount of the DNA. Furthermore, the manual experiments of the affinity separation are certain to involve some errors. Braich et al. used a separation method with a gel-filled tube module to improve the efficiency in the SAT problem [2], and other related works have been done in relation with the micro-reactor [5,9,11,12].

In the previous work [8], we have suggested version space learning with DNA molecules as a concept learning method. This learning method can be implemented by repeated application of the affinity bead separation and this is described in the next section in detail. But, in practical implementation, this method needs much effort to repeat the same procedure, which may also suffer from low yield.

In this paper, we present the design of an affinity separation module, which performs the operation in a single straight channel. The magnetic beads with DNA probe sequences immobilized on their surface were used to select target strands, and the beads are restrained in the channel by a permanent magnet on top of the module. The separation process consists of hybridization and denaturation as in the manual experiment. We have implemented a prototype of this module and some experimental results are presented to verify its working process. This module can be used as well in other applications of the DNA computing which uses affinity separation.

This paper is organized as follows. In Section 2, the workflow of the bead separation for the DNA based concept learning is presented. Section 3 describes the detailed structure of the lab-on-a-chip (LOC) module for bead separation and fabrication process. At the same time some experimental results for this module are presented in Section 4. Finally, the conclusion and future work are given in Section *5*.

2 Concept Learning on LOC

A detailed description on the original version space learning in silico can be found in [10], and the DNA based method is described in [8]. In this section, we explain how a network of the bead separation modules can organize the DNA based concept learning.

Given a training example e, learning proceeds by selecting all hypotheses, which are consistent with e. This process can be achieved through the affinity separation, which examines each attribute value, as is described in Fig. 1. Let T_n be the current version space[1], T_{n+1} the one after a training example e, and T^e a set of hypotheses that classify e as positive. In this paper, we assume three attributes (A, B, C) each of which has only two values. Attribute values are denoted by a lowercase letter with subscripts 1 and 2, and the "don't care symbol" is denoted by subscript 0. Each rectangle in Fig. 1 means an affinity separation module that divides the solution in the manner of whether each hypothesis has at least one of the attribute values, which are denoted in the rectangle. Positive selection means the selection of the DNA strands that hybridize with the bead, and negative selection means the selection of the ones that does not. In principle, the learning process for a single example can be accomplished with this process in Fig. 1.

[1] In this paper, the concept 'version space' is not the one that is maintained by special and general boundaries as in [2], but simply a set of hypotheses that are consistent with training examples.

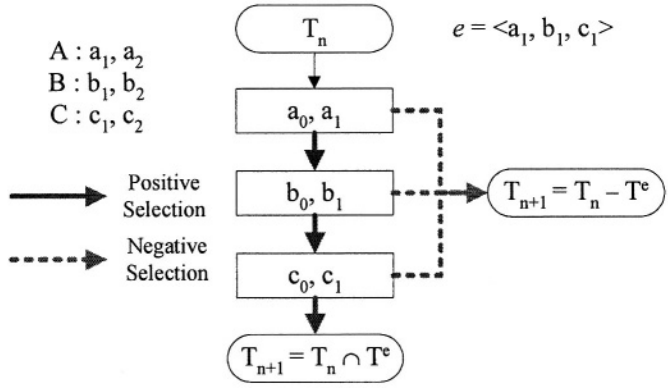


Fig. 1. Training process of DNA based version space learning for an example $<a_1, b_1, c_1>$

However, due to the possibility of false negatives in the selection process, the approach displayed above may be unreliable in the negative training examples. Therefore, this model is modified as follows for reliability, which uses only the positive selection as in Fig. 2.

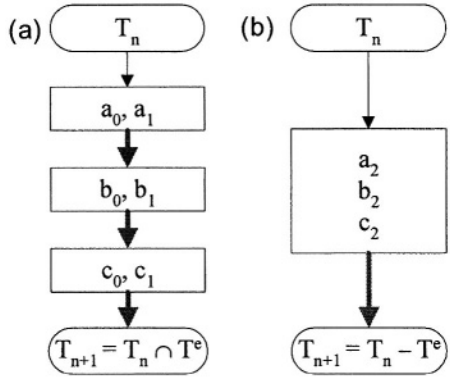


Fig. 2. Modified version of the learning process, which uses only positive selection for an example $<a_1, b_1, c_1>$. (a) is for a positive example, and (b) is for a negative example

Each learning step for a single training example can be implemented by the processes above, and therefore, the full learning process of the DNA based version space learning can be accomplished by networking the above training modules for the training examples.

3 LOC Module for Bead Separation

3.1 Layout of the Module

Basically, the module consists of a single straight channel, which is 2 mm in width, 100 μm in depth, and 44 mm in length. It should be miniaturized. This module uses

streptavidin coated magnetic beads to immobilize the probe sequences of which 5' ends are biotinylated. A permanent magnet is used to immobilize the magnetic beads. It is on top of the module and we can control the bead to flow or to make immobilized by the magnet.

To perform affinity separation, we need two steps, i.e. hybridization and denaturation as in the manual affinity separation. In the hybridization step, we first injected the mixed solution through the channel. This gave a rise to hybridize both target and probe DNA strands on the magnetic beads. After then, 0.1 M NaOH solution through the channel and the denaturized DNA strands pass through the channel.

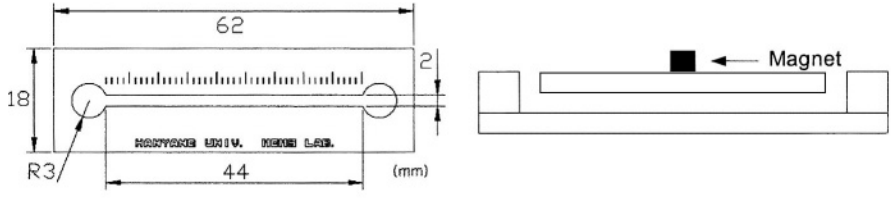


Fig. 3. Layout of the bead separation module. Top view (left) and side view (right)

3.2 Fabrication

The negative photo resistant SU-8 (Micro Chem, USA) was spun coated on the substrate 100 Si wafer (100 μm thick). And the air bubble of the coated SU-8 was removed by soft baking on a hot plate (65°C for 10 min, 95°C for 30 min). After installing and hardening the mask on the wafer, photolithography was performed on the MA-6 aligner. Finally this was cleaned with isopropyl alcohol after development by SU-8 developer. PDMS (polydimethylsiloxane, Corning, Sylgard 184) and curing agent were mixed (ratio of 10:1) and cured in oven (65°C, 4 h). PDMS replica was peeled off the mold, treated in O_2 plasma and bonded with a slide glass (Corning, #7740 Pyrex glass).

Two holes of 2 mm in diameter were drilled in the center of each reservoir port for inlet and outlet of solution. A magnet (CPG mini separator) seated on top of the device over channels was used to localize the beads in the detection area. It was ø 6 mm × 5 mm with strength 12,200 Gauss 1,220 mTesla at the pole.

The summary of the above process is shown in Fig. 4.

4 Experimental Results

In this section, we describe some experimental results for the above module. Firstly, the magnetic beads with the probe sequences were prepared, which were then installed into the module. We confirmed the separation by FITC modified strands, as well as measuring the yield by UV spectrophotometer and then compared these results with those of the manual process.

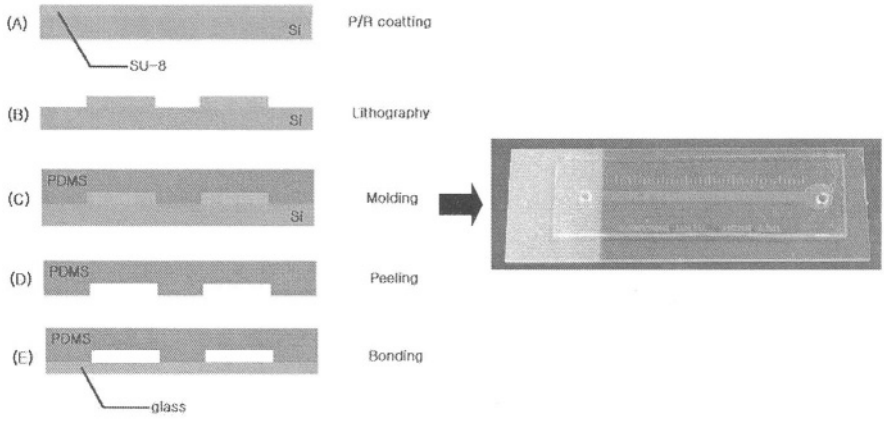


Fig. 4. Fabrication process and final module

4.1 Apparatus

Reagents were loaded into the fluid lines (silicon tube ø 1 mm) with 250 μl, ø 2.30 mm Hamilton syringe and then pushed through the device syringe pump (KD scientific, AUXETAT). The device was mounted on an inverted fluorescence microscope for detection (NIKON, DIAPHOT 300). The microscope consisted of a halogen lamp for top illumination and a 100 W xenon lamp for sample illumination from the bottom. A charge coupled device (CCD) camera (IK-642F, Toshiba) was used to monitor the transport of magnetic beads and to measure fluorescence intensity. An interference filter cube was used to detect fluorescence signals. The bandwidth of the excitation filter was 459~498 nm and 512~559 nm for the emission filter. The images of the beads and fluorescence were acquired and analyzed with imaging software (Image tool 3, UTHSCSA).

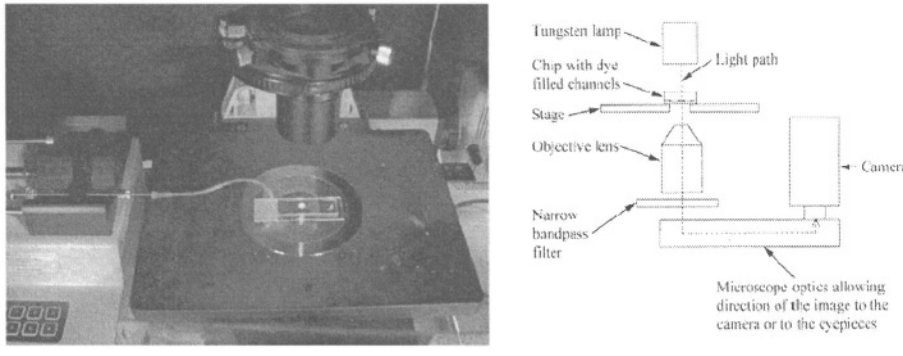


Fig. 5. Apparatus

4.2 Oligonucleotides

We used the sequence generator NACST/Seq [7] to design the DNA sequences for the experiment. When designing the oligonucleotide sequences, we considered the self-

homology and the H-measure to prevent cross hybridization. We designed three sequences of which 20 bp of 3' ends were selected as target subsequences and the probe sequences were determined from these subsequences. Table 1 shows the six sequences. All oligonucleotides were purchased from GenoTech (Daejeon, Korea) and obtained 5' FITC modified target strands for experiments to confirm the separation in the module in addition to the original three target strands. As well the probe sequences (No. 4~6 in Table 1) were 5' biotinylated for immobilization on the magnetic beads. As a result, we synthesized nine oligonucleotides for the experiment.

Table 1. Sequences of oligonucleotides used in the experiment

No.	Sequence	Tm (°C)	GC%
1	5'-AAACGGTCTTGTCGG **GGATAGGGAATGCTCGTGTT**-3'	70.1	51.4
2	5'-GTCGAGACAAGACCT **TGGCATATGTCTAAATAGAT**-3'	67.1	40.0
3	5'-GCGAATCTGATCGCT **CCAGCGAGGGCCGGTTCCAC**-3'	77.7	65.7
4	5'- AACACGAGCATTCCCTATCC -3'	57.3	50
5	5'- ATCTATTTAGACATATGCCA -3'	49.1	30
6	5'- GTGGAACCGGCCCTCGCTGG -3'	67.6	75

4.3 Probe Bead Preparation

Conjugation of biotinylated single-stranded DNA to M-280 beads (Dynal M-280 streptavidin coated, ø 2.8 μm) was performed by using the following protocol. At first, 300 μl of stock beads ($6\sim7 \times 10^8$ bead/Ml) were washed three times in 200 μl of Binding and Washing (B/W) buffer which consisted of 10 mM Tris, 1 mM EDTA, and 1 M NaCl, pH 7.5 and then diluted in 90 μl of B/W buffer. And 10 μl, 100 μM biotinylated DNA solution was added to the beads solution and then incubated at room temperature for 15 min. After conjugation, the beads were washed with B/W buffer and diluted in 100 μl of TE buffer, which was made up of 10 mM Tris, 1 mM EDTA, pH 8.0.

4.4 Separation

In this experiment, oligonucleotide No. 2 in Table 1 was the target strand, the mixed solution, which contained oligonucleotides No. 1~3 were used in separation.

Before the experiments, the module and the other devices were washed with ultra sonicator (Branson) for 5 min and 0.1 M NaOH. Then the inlet port of the module was connected to the syringe pump. TE buffer was mechanically pumped into the silicon tube (ø 1 mm) and flushed the device to remove out air bubbles. The dead volume in the tube and the connectors was 10 μl in case of 250 μl Hamilton syringe.

Detection of Separation with FITC Modified Oligonucleotides

The solution that contained probe beads (30 μl of No. 5 probe attached bead + 70 μl of TE buffer) was injected into the channels by using syringe pump at the velocity of 20 μl/min. The mixed solution which contained each 10 μl of No. 1~3 that are FITC modified and 170 μl TE buffer, was injected into the channel by 5 μl/min for 40 min.

After hybridization, unbounded oligonucleotides were washed with 100 μl TE buffer by 10 μl/min, 2 times, and the intensity of fluorescence signal was detected by CCD camera. For elution of the bound target strands, 200 μl of 0.1 M NaOH was injected into the channel (10 μl/min).

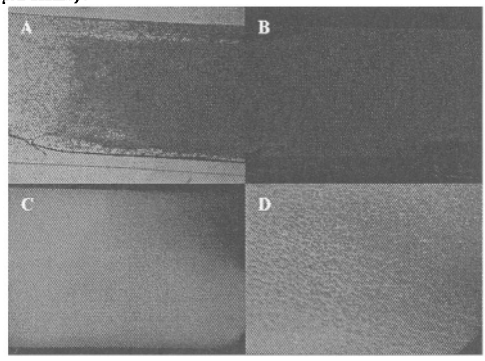


Fig. 6. Normal and fluorescent images of the beads in the module. Beads are localized in a channel by a magnet (a) Injection of M-280 bead in channel, 40X under halogen lamp, (b) 40X fluorescence image before hybridization with FITC conjugated probe, (c) 40X fluorescence image after an injection of the mixed solution which contains FITC modified target strands, (d) 100X fluorescence image, black dots are M-280 beads, green background is target solution

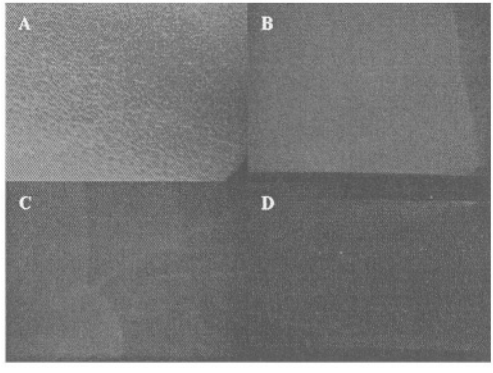


Fig. 7. Fluorescence images in washing and elution process (a) Before washing, (b) After washing, (c) Elution step: green narrow flow indicates the target oligonucleotide, (d) After completing the elution step

Figs. 6 and 7 show the separation process in the module. We can observe that the probe beads capture the target strands in hybridization step and that the strands are eluted in denaturation step.

Detection with UV Spectrophotometer

In the previous subsection, the solution, which contains the probe beads and the mixed solution were injected into the channel step by step, except that the mixed solution consisted of unmodified oligonucleotides. The waste solution of the

hybridization step containing unbounded strands was collected and then optical density of this solution was measured at 260 nm by UV spectrophotometer (UV-1601, SHIMADZU). The waste solution in the elution process was collected and optical density was measured to compare intensity profile before and after the separation.

In addition to this, we performed a manual separation process, using the same protocol without a separation module to compare the efficiency.

Table 2. Optical density of the solution at each step

Concentration	*Module*	*Manual*
Initial mixed solution	184.7 μg/ml	184.7 μg/ml
Waste in hybridization	164.8 μg/ml	146.1 μg/ml
Waste in denaturation	27.8 μg/ml	17.88 μg/ml

Table 2 is the result of the separation and the amount of the DNA strands in each step. As shown in the table, the yield is better than that of the manual process although the module misses out the much of the target strands.

5 Conclusion

In this work, we have shown that the DNA based concept learning can be performed by iterative affinity separations and also its concrete workflow has been presented. The learning process purely consists of the affinity separations, which examine each attribute value of the hypotheses in the current version space. Therefore, the network of the affinity separation modules can implement this learning scheme. For this purpose, we designed and implemented a prototype of a single LOG module for the affinity bead separation. The separation process was verified by several experiments, and the results show the separation process is performed more correctly and efficiently than manual work.

The yield of the affinity separation process can be improved further. The design or the strategy of the separation module needs to be improved to increase the efficiency of the probe beads. Undoubtedly, there may be other methods of immobilizing the probe DNA or the magnetic bead. Nevertheless, it is important that the probe sequences should be able to have more chances to hybridize with demanded target strands, and after all the proportion of the false negatives must be decreased as a result. The affinity separation used in [3] may be a good example. On the other hand, we may use the negative selection strategy to overcome the very low yield of the positive selection or the error of the positive selection in combination with a probability theory. However, we need a quantitative analysis of this process to support it. More sophisticated research about this affinity separation must be performed.

And more miniaturization, integration and automation are required for the real application. For example, Choi et al. have developed a technology to separate the magnetic beads in a micro-channel by electromagnet [4]. We may as well be able to control the beads in the module in combination with this method.

Acknowledgement

This research was supported by the Ministry of Commerce, Industry and Energy through the Molecular Evolutionary Computing (MEC) project, the NRL program from Korean Ministry of Science and Technology, and the Ministry of Education & Human Resources Development under the BK21-IT program. The ICT at Seoul National University provided the research facilities for this study.

References

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems, *Science,* vol. 266, pages 1021-1024, 1994

[2] Eric B. Baum. Building an associative memory vastly larger than the brain. *Science,* vol. 268, pages 583-585, April 28, 1995

[3] R. S. Braich, N. Chelyapov, P. W. K. Rothermund, and L. Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer, *Science,* 295, pages 499-502, 2002

[4] J.-W. Choi, T. M. Liakopoulos, C.-H, Ahn, An on-chip magnetic bead separator using spiral electromagnets with semi-encapsulated permalloy, *Biosensors & Bioelectronics,* vol. 16, pages 409-416, 2001

[5] Z. H. Fan, S. Mangru, R. Granzow, P. Heaney, W. Ho, Q. Dong, and R. Kumar, Dynamic DNA hybridization on a chip using paramagnetic beads, *Anal. Chem.,* vol. 71, pages 4851-4859, 1999

[6] J. Khodor, and D. K. Gifford, The efficiency of sequence-specific separation of DNA mixtures for biological computing, *3rd Annual DIMACS Workshop on DNA Based Computers,* Philadelphia, Pennsylvania, June, 1997

[7] D.-M. Kim, S.-Y. Shin, I.-H. Lee, and B.-T. Zhang, NACST/Seq: A sequence design system with multiobjective optimization, *DNA8 Lecture Notes in Computer Science,* vol. 2568, pages 242-251, 2003

[8] H.-W. Lim, J.-E. Yun, H.-M. Jang, Y.-G. Chai, S.-I. Yoo, and B.-T. Zhang, Version space learning with DNA molecules, *DNA8 Lecture Notes in Computer Science,* vol. 2568, pages 143-155, 2003

[9] J. S. McCaskill, R. Penchovsky, M. Gholke, J. Ackermann, and T. Rücker, Steady flow micro-reactor module for pipelined DNA Computations, *Proceedings of 6th International Meeting On DNA Based Computers,* pages 239-246, 2000

[10] T. M. Mitchell, *Machine Learning,* 1997, McGraw-Hill

[11] D. van Noort, F.-U. Gast, and J. S. McCaskill, DNA computing in microreactors, *Proceedings of 7th International Meeting on DNA Based Computers,* pages 128-137, 2001

[12] R. Penchovsky, and J. S. McCaskill, Cascadable hybridisation transfer of specific DNA between microreactor selection modules, *DNA7 Lecture Notes in Computer Science,* vol. 2340, pages 46-56, 2002

Parallel Translation of DNA Clusters by VCSEL Array Trapping and Temperature Control with Laser Illumination

Yusuke Ogura[1,3], Takashi Kawakami[1], Fumika Sumiyama[1,3], Akira Suyama[2,3], and Jun Tanida[1,3]

[1] Graduate School of Information Science and Technology
Osaka University
2-1 Yamadaoka, Suita, Osaka 565-0871, Japan
{ogura,kawakami,sumiyama,tanida}@ist.osaka-u.ac.jp
[2] Graduate School of Arts and Sciences
The University of Tokyo
3-8-1 Komaba, Meguro-ku, Tokyo 153-8902, Japan
suyama@dna.c.u-tokyo.ac.jp
[3] Japan Science and Technology Corporation (JST-CREST)

Abstract. This paper reports an experimental verification on the fundamentals of a translation method of DNA clusters. The DNA cluster is composed of DNA strands connected to a microscopic bead. We demonstrated that the DNA clusters were translated toward different directions by optical manipulation with vertical-cavity surface emitting laser (VCSEL) array sources. The DNA strands connected to the target bead by hybridization were detached by temperature control with laser illumination. The presented method is expected to be applied to DNA computing and molecular memory that utilize optical techniques effectively.

1 Introduction

Ashkin *et al.* have demonstrated optical levitation of a particle by scattering force in 1971 [1] and three-dimensional trapping of a microscopic dielectric particle by optical gradient force in 1986 [2]. An optical manipulation technique, which utilizes radiation pressure force induced by illumination of a light wave onto an object, is a useful tool for a wide variety of fields of science. Proper control of the radiation pressure force provides us a noncontact and accurate manipulation method on microscopic objects. Applications of optical manipulation cover transport of organelles in living cells [3], an optical spin micro-motor [4], optical alignment and spinning of birefringent fragments [5], and so on.

It is important to generate an appropriate light field for achievement in an advanced manipulation of objects. Use of the Bessel beam of light makes it possible to manipulate particles in multiple planes simultaneously even though

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 10–18, 2004.

the light beam is partially blocked[6], Spatial and temporal modulation of a single light beam is also effective to arrange particles on a plane and to fabricate three-dimensional structures [7].

We have proposed a new type of an optical manipulation technique with a vertical-cavity surface-emitting laser (VCSEL) array as light sources [8, 9]. We call the technique VCSEL array trapping. The VCSEL array is an array of semiconductor laser sources arranged in several tens or hundreds micrometer period on a substrate. Emission intensities of the individual pixels of the VCSEL array are independently controlled by electronics with gigahertz order of the maximum modulation rate. The features of the VCSEL array can be utilized to construct an optical manipulation system that is capable of parallel and flexible manipulation based on compact hardware and a simple control method. We have demonstrated some functions of the VCSEL array trapping such as parallel translation of particles without mechanical equipment, optical levitation of a particle by illumination of multiple beams, and stacking of particles[8, 9, 10].

This paper focuses on an experimental verification of a translation method for DNA clusters by the VCSEL array trapping and temperature control with laser illumination. Although direct manipulation of a molecule by use of radiation pressure force is difficult, indirect manipulation is a practical method for the purpose. For example, if target molecules are attached to a microscopic bead, the molecules can be dealt with by optical manipulation. The same technique is applied to measure the shearing force of a single knotted DNA molecule[11], to clarify a mechanism of the motility of motor proteins[12], and so on.

A bead, on which DNA strands with an anti-tag sequence are immobilized by the biotin-streptavidin bond, is introduced to translate a DNA cluster. The DNA strands in the cluster include a tag sequence at a terminal and can be attached to the bead by hybridization with the DNA strands that include the anti-tag sequence. In our method, the DNA cluster is attached to and detached from a specific bead by changing the temperature around the bead with laser illumination.

A control method for positioning and the reaction of DNA molecules at local spaces is provided by utilizing the optical techniques effectively. This method is expected to open a door for a new class of molecular computing, molecular memory, and other fields [13].

2 Parallel Translation Method of DNA Clusters

The conceptual diagram of the VCSEL array trapping is shown in Fig. 1. Flexible manipulation for micro-objects is achieved by control of spatial and temporal intensity distribution generated by VCSEL array sources. The VCSEL array trapping provides many benefits. Notable points are as follows. (1) Parallel manipulation of multiple objects is straightforward. (2) The VCSEL array is easily combined with micro-optics. For example, a board-to-board free-space optical interconnect is realized by a VCSEL array and microlenses without external relay optics [14]. This fact suggests that the VCSEL array has potential capability to

reduce hardware complexity. (3) No control other than modulating the emission intensities of the VCSELs is required because additional devices for a specific manipulation are not necessary. Therefore, a troublesome control method can be avoided. (4) Various modes of manipulations are achieved by the same system configuration. These respects are helpful to manipulate DNA clusters.

The scheme of a translation method of DNA clusters utilizing optical techniques is shown in Fig. 2. This method is based on a parallel translation technique by the VCSEL array trapping and temperature control with laser illumination.

If the DNA strands are dispersed in a solution, it is difficult to make a DNA cluster by using radiation pressure force. Therefore, micro beads are used to overcome the problem. A lot of DNA strands with a special base sequence called anti-tag sequence are immobilized to the surface of the beads. The target DNA strands include a complement sequence (called a tag sequence) of the anti-tag sequence. After reaction of the target DNA strands and the beads, the target DNA strands are attached to the beads by hybridization of the tag and the anti-tag sequences. As a result, the DNA clusters composed of many target DNA strands are produced for the individual beads. Then the DNA clusters can be translated toward different directions simultaneously by control of the emission pattern of the VCSEL array.

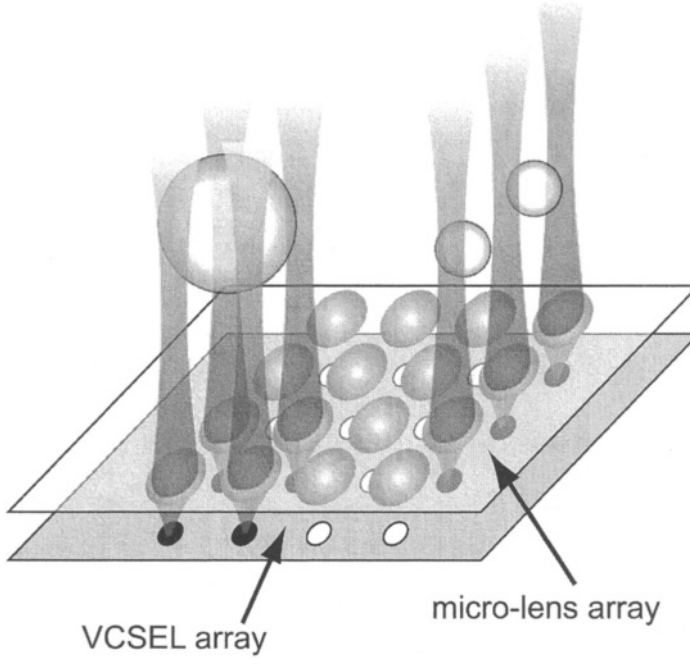


Fig. 1. The conceptual diagram of VCSEL array trapping

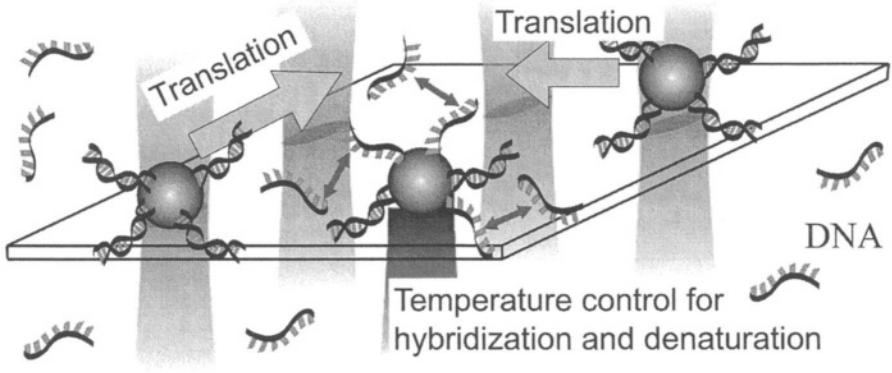


Fig. 2. A translation method of DNA clusters by VCSEL array trapping and temperature control

Attaching a DNA cluster to a bead and detaching it from the bead are achieved by temperature control in local space with laser illumination as shown in Fig. 3. A DNA solution is on a substrate whose surface is coated by a sort of material that absorbs light. The substrate is heated up by illuminating a focused laser beam due to absorption of light. Thermal energy transfers to the solution on the substrate, then the temperature of the solution around the illumination area becomes higher than that of the solution far from the illuminated area. Based on this phenomenon, the local temperature of the solution can be controlled by

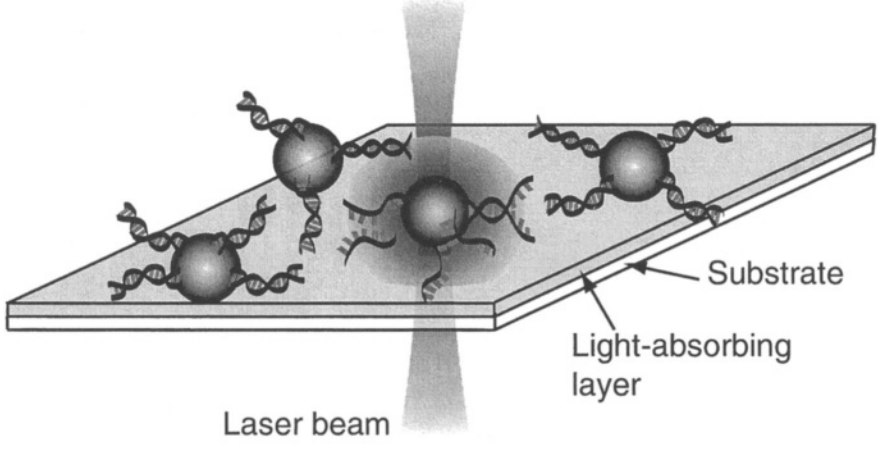


Fig. 3. An optical method to control reaction of DNA molecules in local space

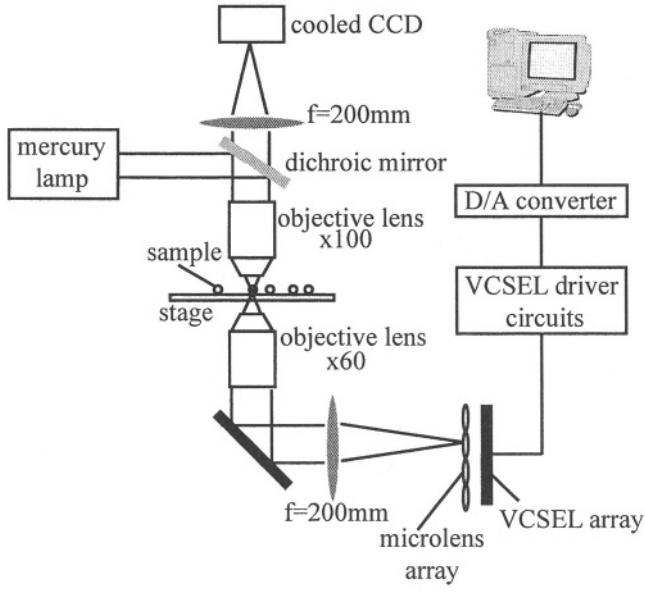


Fig. 4. Experimental system of VCSEL array trapping

increasing or decreasing the power of the illumination beam. With a well-focused beam, reaction of the DNA cluster is controlled selectively.

3 Experimental Results

Translation of a DNA cluster includes three steps of operations: (i) attaching the DNA cluster to a bead, (ii) translation of the bead, and (iii) detaching the DNA cluster from the bead. Unfortunately, we have not verified step (i) yet, so we describe experimental verification of steps (ii) and (iii) in this paper.

Figure 4 shows an experimental setup to demonstrate parallel translation of the DNA clusters (step (ii)). A VCSEL array (NTT Photonics Laboratory, wavelength of 854nm±5nm, maximum output power of more than 3mW, aperture of 15μm-diameter, and pixel pitch of 250μm) has 8×8 VCSEL pixels, after which a micro-lens array (focal length of 720μm and lens pitch of 250μm) was set to increase light efficiency. Emission intensities of the VCSELs were controlled by a personal computer (DELL; 800MHz PentiumIII processor) through voltage-to-current conversion circuits. A water-immersible, long working-distance objective lens (OLYMPUS, LUMPlan Fl 60× W/IR, NA=0.90) was used as the focusing lens. The beam spacing period, the maximum intensity, and the beam diameter on the sample plane were 3.75μm, approximately 1.1mW per pixel, and 2.6μm, respectively. The sample plane was observed by the cooled CCD (Nippon Roper, CoolSNAP fx).

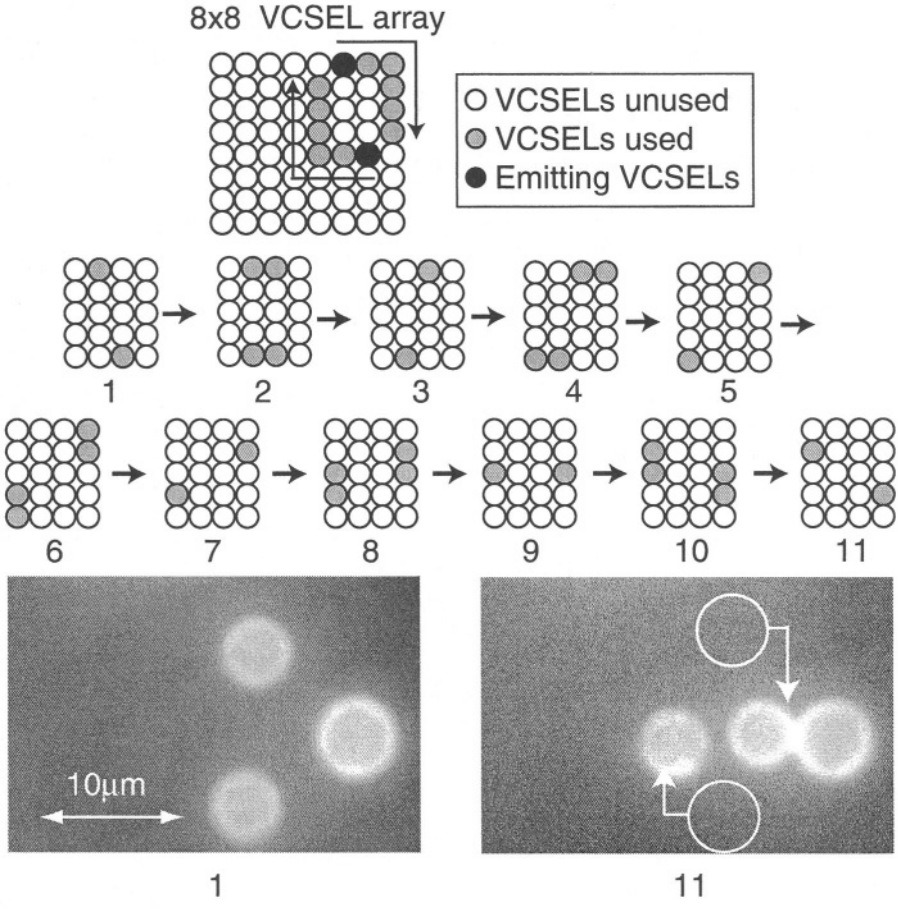


Fig. 5. Experimental result of parallel translation of DNA clusters. Upper; a sequence of emission pattern of the VCSEL array, and lower; fluorescent images before (left) and after (right) translation

The beads used in the first experiment were polystyrene particles of 6μm in diameter (Polysciences, Inc., Streptavidin Coated Carboxylated Microspheres) whose surface was coated by streptavidin. First, DNA strands with an anti-tag sequence (3'-GCACCTAGTCATTGACTTTACTCCATTCTAAACATGATAC-5', T_m = 63℃) were immobilized to the beads by biotin-streptavidin binding. Second, fluorescent molecules (Molecular Probes: Alexa Fluor 546) were attached to the target DNA strands with a tag sequence(3'-GTATCATGTTTAGAATGG AGTAAAGTCAATGACTAGGTGC-5') to sense the DNA strands for observation. Third, the target DNA strands were mixed to the solution of the beads, and attached to the beads by hybridization. Then the beads were extracted and mixed into TE buffer solution (pH=8.0).

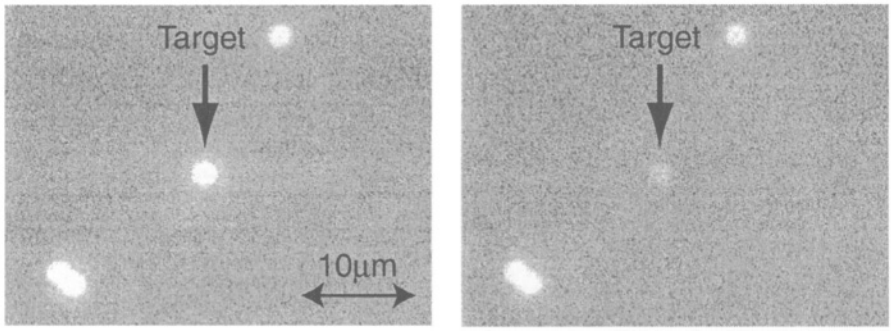


Fig. 6. Observed fluorescent images (left) before and (right) after 3 cycles of illumination

Figure 5 shows an experimental result on simultaneous translation of two DNA clusters. The upper part is a sequence of emission patterns of the VCSEL array and the lower is the observed fluorescent images. One or two pixels of the VCSEL array were assigned to capture an individual bead in this experiment. This result demonstrated that the beads (DNA clusters) were translated toward different directions in response to the emitting pixels simultaneously. The translation path length was $19\mu m$ and the average velocity was $0.38\mu m/sec$.

The next experiment is detachment of the DNA clusters from the bead by laser illumination (step (iii)). A glass substrate is coated by titanylphthalocyanine with the thickness of $0.15\mu m$. The beads used in this experiment were particles of $2\mu m$ in diameter. The DNA clusters were prepared by the same procedure for the experiment of parallel translation. A sample solution was sandwiched between the substrate and a cover slip. A particular area on the substrate where the bead existed was illuminated by a focused beam emitted from a He-Ne laser (wavelength: 633nm). We illuminated the area for 15 second then stopped illuminating for 4 second to capture a fluorescent image during one cycle. In the experiment, this illumination cycle was repeated.

Figure 6 shows fluorescent images before and after 3 cycles of illumination with power of 3.5 mW. As seen from the figure, the fluorescent intensity of the target bead decreases considerably. In contrast, when we used a pure substrate instead of that coated by titanylphthalocyanine, the fluorescent intensity of a bead decreased little during illumination. This means that decrease of the intensity is not lead by photobleach of the fluorescent molecules. We can conclude that the DNA cluster is detached from the bead due to the local rise of the temperature. Note that the fluorescent intensities of the beads, which were approximately $10\mu m$ distant from the target bead, showed only a little decrease. Therefore, the area where the temperature is affected is considered to be restricted within about $10\mu m$ from the illumination point under the condition examined.

We also investigated the decay speed of the fluorescent intensity. The fluorescence from the target bead approximately disappeared after 10 cycles of

illumination with the power of 3.5 mW. On the other hand, with the power of 2.0 mW, the fluorescence can be clearly observed after 10 cycles. The fluorescent intensity is related to the number of the DNA strands attaching to the bead. Consequently, this result suggests that the detaching rate can be controlled by illumination power.

4 Potential Applications

The experimental results suggest the potential capabilities of the optical techniques for control of the position and reaction of DNA molecules. Although our experiments demonstrated only the fundamentals of the method, future refinement is expected to contribute to various scientific fields. In particular, parallel information processing that uses DNA molecules such as DNA computing and molecular memory are interesting applications of the presented method. In the experiment, the whole sequence of the target DNA strands is the tag sequence, so that the sequences of all DNA strands are exactly the same. However, an additional sequence, which corresponds to coded data in DNA computing, can be appended to the target DNA sequence. In this case, DNA clusters contain DNA molecules with various base sequences for individual beads. With the bead, different data can be manipulated simultaneously. Furthermore, a capability to control the DNA reaction in local space can be used for distributed processing with the DNA molecules, which increases the flexibility of operations.

5 Summary

A new optical method was studied to control the position and the reaction of DNA clusters. We succeeded in parallel translation of DNA clusters by the VCSEL array trapping. Detaching DNA from a bead was also demonstrated by temperature control in local space with laser illumination. These results suggest that optical techniques are useful in control of DNA molecules. In future, a new class of parallel computing paradigm is expected to be developed by using both light and DNA as an information carrier based on the presented method.

Acknowledgments

This work is supported by JST CREST.

References

[1] A. Ashkin and J. M. Dziedzic: Optical levitation by radiation pressure. Appl. Phys. Lett. **19** (1971) 283–285 10

[2] A. Ashkin, J. M. Dziedzic, J. E. Bjorkholm, and S. Chu: Observation of a single-beam gradient force optical trap for dielectric particles. Opt. Lett. **11** (1986) 288–290 10

[3] A. Ashkin, J. M. Dziedzic, and T. Yamane: Optical trapping and manipulation of single cells using infrared laser beams. Nature **330** (1987) 769–771 1.0

[4] Z. P. Luo, Y. L. Sun, and K. N. An: An optical spin micromotor. Appl. Phys. Lett. **76** (2000) 1779–1781 10

[5] M. E. J. Friese, T. A. Nieminen, N. R. Heckenberg, and H. Rubinsztein-Dunlop: Optical alignment and spinning of laser-trapped microscopic particles. Nature **394** (1998) 348–350 10

[6] V. Garces-Chavez, D. McGloin, H. Melville, W. Sibbett, and K. Dholakia: Simultaneous micromanipulation in multiple planes using a self-reconstructing light beam. Nature **419** (2002) 145–147 1.1

[7] M.P. MacDonald, L. Paterson, K. Volke-Sepulveda, J. Arlt, W. Sibbet, K. Dholakia: Creation and manipulation of three-dimensional optically trapped structures. Science **296** (2002) 1101–1103 11

[8] Y. Ogura, K. Kagawa, and J. Tanida: Optical manipulation of microscopic objects by means of vertical-cavity surface-emitting laser array sources. Appl. Opt. **40** (2001) 5430–5435 11

[9] Y. Ogura, N. Shirai, and J. Tanida: Optical levitation and translation of a microscopic particle by use of multiple beams generated by vertical-cavity surface-emitting laser array sources. Appl. Opt. **41** (2002) 5645–5654 11

[10] F.Sumiyama, Y.Ogura, and J.Tanida: Stacking and translation of microscopic particles by means of 2ü 2 beams emitted from vertical-cavity surface-emitting laser array. Appl. Phys. Lett. **82** (2003) 2969-2971 11

[11] Y. Arai, R. Yasuda, K. Akashi, Y. Harada, H. Miyata, K. Kinosita Jr., and H. Itoh: Tying a molecular knot with optical tweezers. Nature **399** (1999) 446–448 11

[12] C. Veigel, L. M. Coluccio, J. D. Jontes, J. C. Sparrow, R. A. Milligan, and J.E. Molloy: The motor protein myosin-I produces its working stroke in two steps. Nature **398** (1999) 530–533 11

[13] L. M. Adleman: Molecular computation of solutions to combinatorial problems. Science **266** (1994) 1021–1024 11

[14] E.M. Strzelecka, D. A. Louderback, B.J. Thibeault, G. B. Thompson, K. Bertilsson, and L. A. Coldren: Parallel free-space optical interconnect based on arrays of vertical-cavity lasers and detectors with monolithic microlenses. Appl. Opt. **37** (1998) 2811–2821 11

Chemical Switching and Molecular Logic in Fluorescent-Labeled M-DNA

Shawn D. Wettig, Grant A. Bare, Ryan J. S. Skinner, and Jeremy S. Lee

Department of Biochemistry, University of Saskatchewan,
107 Wiggins Rd., Saskatoon, SK S7N 5E5, Canada
{Wettig,Leejs}@sask.usask.ca

Abstract. M-DNA is a complex formed between duplex DNA and divalent metal ions at approximately pH 8.5. 30 base pair linear duplexes were prepared with fluorescein attached to one end, and various electron acceptors at the other. Quenching of the fluorescence emission from fluorescein by the acceptor molecules was observed under conditions corresponding to M-DNA, but not for B-DNA. For the case of anthraquinone as the acceptor, the quenching, which is ascribed to an electron transfer process, was blocked by chemical reduction ($NaBH_4$) of anthraquinone to the dihydroanthraquinone which is not an electron acceptor. Upon the reoxidation of the dihydroanthraquinone by exposure to oxygen the quenching was restored. Quenching of fluorescein fluorescence was also observed in a 90 base pair Y-branched duplex in which rhodamine or anthraquinone were attached to one or two of the remaining arms. Thus the electron transfer process is not impeded by the presence of a junction in the duplex, contrary to results previously reported for B-DNA samples. Again the fluorescein fluorescence could be modulated by reduction of the anthraquinone group in the Y-branched duplexes, mimicking a simple chemical switch. A number of molecular logic functions are demonstrated in M-DNA by considering various chemical inputs, with the level of observed quenching serving as the observed output. Therefore M-DNA may have extraordinary potential for the development of nano-electronic devices.

1 Introduction

The continued demand for increased speed and processing capabilities in micro-computers has led to a significant increase in research in the design of nano-electronic devices; in particular research into materials suitable for the manufacturing of nano-electronic devices. DNA is of particular interest, in light of its molecular recognition and self-assembling properties as well as the possibility of allowing charge transfer. Indeed, the self-assembly capability of DNA has been used to demonstrate a number of nano-mechanical devices. For example, Seeman et al. have recently reported a number of devices which take advantage of double [1] and paranemic [2,3] crossovers to bring about substantial changes in DNA conformation. These changes form the basis of mechanical switches. Other devices, recently reviewed by Niemeyer et al., take advantage of DNA conformational effects resulting from changes in metal ion concentration, and ultra- and intermolecular hybridization [4].

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 19-31, 2004.

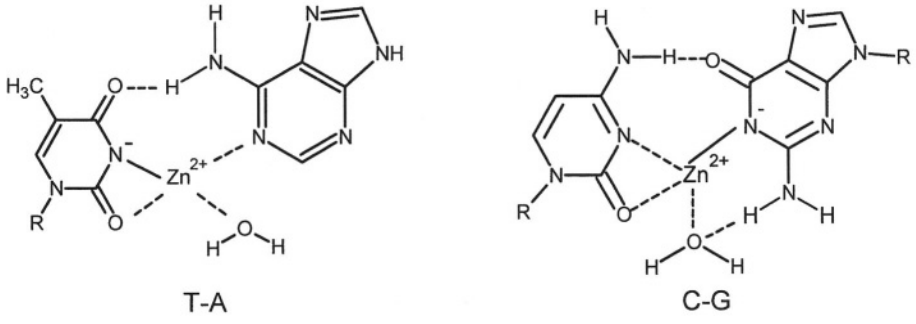


Fig. 1. Proposed base-pairing in M-DNA. The metal ion replaces the imino proton of *T* and *G*. Adapted from reference [14]

Of course a more critical aspect of the application of DNA to the development of nano-electronic devices is its ability to conduct electrons, i.e., its charge transfer capabilities. There is significant controversy with respect to whether or not DNA can provide an efficient pathway for electron transfer to occur. It has been suggested that the stacked aromatic bases of the DNA duplex provides an efficient means of electron transfer (the so-called π-way)[5-7]; however, recent reports show B-DNA to behave as a semi-conductor[8-10] or even as an insulator.[11] It has been suggested that the DNA duplex could be coated with a thin film of metal atoms to improve conduction,[12,13] but unfortunately this destroys the desirable molecular recognition properties of DNA.

Our solution to this problem is based upon a novel DNA-metal ion complex discovered in our lab known as M-DNA. M-DNA is a complex in which a divalent metal ion (Zn^{2+}, Ni^{2+}, or Co^{2+}) is incorporated into the center of the DNA duplex under specified conditions of pH and metal ion concentration. [14-16] The addition of metal ions at elevated pH results in the deprotonation of the N3 or Nl position of T (pK_a 9.9) or G (pK_a 9.4), respectively. This results in a series of aligned metal complexes that has the effect of improving electron transfer and increasing the overall conductivity of DNA creating, in effect, a molecular wire. The deprotonation of the N3 and Nl positions of T and G has been observed using pH titration[14] and NMR[16] measurements, the results of which have served as a basis (along with circular dichroism results) for the proposed structure obtained from molecular modeling studies (Figure 1).[14] Crystallographic and X-ray spectroscopic studies are currently underway to confirm this structure.

The ability of M-DNA to serve as a conductor of electricity has been demonstrated by direct measurements of conductivity. Briefly, B- and M-DNA duplexes were placed across a deep gap between two electrodes and the current-voltage characteristics were measured. [9] Semi-conducting behavior (with a narrow band-gap) was observed for B-DNA, while M-DNA showed metallic-like conduction. Efficient electron transfer has also been demonstrated using fluorescence quenching measurements. Linear duplexes of 20 base pairs were labeled with fluorescein and rhodamine at opposite ends, and upon M-DNA formation quenching of the fluorescein fluorescence was observed. Analysis of fluorescence lifetime data

indicated an electron transfer mechanism for this quenching, observed only under M-DNA conditions.[14] Distance dependent studies have shown efficient electron transfer in duplexes up to 500 base pairs (~150 nm) in length, with the distance dependence of rate of electron transfer supporting a hopping mechanism for electron-transfer in M-DNA.[15]

In this work we report the results of a study of fluorescence quenching of the electron donor fluorescein, by electron acceptors such as anthraquinone, rhodamine, and Cy5 in M-DNA using linear 30 base pair and Y-branched 90 base pair duplexes. Anthraquinone [17-20] (and derivatives thereof) has been extensively used to probe charge transfer processes occurring in DNA, with the dye (in its excited state) serving as an electron acceptor from guanine; however, they have not been studied in donor/acceptor combinations separated by a DNA duplex. These dyes, and related biologically important quinones, are of particular interest in light of their intimate involvement in electron transport and in the photosynthetic pathway, and are being studied in the development of photosynthetic mimics. [21] Here, the fluorescence of fluorescein is quenched by anthraquinone in M-DNA but not B-DNA. Reduction of anthraquinone to dihydroanthraquinone results in significantly reduced quenching. In effect the electron transfer process is blocked by chemical reduction of the acceptor group. The Y-branched structure is of particular importance in demonstrating not only that specific architectures can be constructed using the self-assembling properties of DNA, but also that electron transfer is not impeded in such structures. Previous studies have shown the branched duplexes to be a Y-shaped molecule with the three arms in an essentially planar geometry with equal angles between each arm.[22,23] The addition of metal cations does not result in helix-helix stacking observed in 4-way junctions, rather the 3-way junction remains in an extended Y-shaped conformation.[22,23] Such junctions, in B-DNA, typically exhibit less efficient electron transfer; [24-26] however, in this work efficient electron transfer is observed to occur between fluorescein and the acceptors anthraquinone, rhodamine, and Cy5 through a Y-branched junction.

The ability to chemically reduce the anthraquinone dye forms the basis of a chemical switch, and the resulting system can be shown to behave as a molecular logic device. This combined with the demonstrated electron transfer through a branched junction, allowing for the possible design of specific molecular architectures, illustrates the enormous potential for the application of M-DNA to the development of nano-electronic devices.

2 Experimental

To evaluate the efficiency of anthraquinone (AQ) as a quencher for fluorescein, measurements were initially carried out using a 30 base pair sequence with the donor strand being labeled in the 5' position with 5-carboxyfluorescein (Fl) and the complementary (acceptor) strands labeled in the 5' position with 2-anthraquinonecarboxylic acid (AQ), 5-carboxytetramethyrhodamine (Rh), pyrenebutanoic acid (PBA), Cy5 or Cy 5.5. The dye molecules were covalently attached using a standard 6-aminohexyl linker. Where necessary, carboxylic acid derivatives were converted to activated esters prior to attachment. Oligonucleotides

were obtained from either the Calgary Regional DNA Synthesis Facility or from the DNA/Peptide Synthesis Lab at the National Research Council Plant Biotechnology Institute (Saskatoon). The oligonucleotide sequences are listed in Table 1.

Three 60 base single-strands were used to form a duplex, of a 90 base pair overall size, containing a Y-junction, allowing for a number of donor-acceptor combinations. Sequences shorter than 60 bases were not used in the preparation of the Y-junctions in order to reduce the probability of fluorescence resonance energy transfer (FRET). The sequences used in this study are given in Table 1 and were labeled in the same manner as the 30 base pair duplexes described above. The Y-junctions were prepared by incubating the three single strands in the dark, in 10 mM Tris-HCl (pH 8) and 10 mM NaCl at 65°C for two hours, followed by slow cooling to room temperature [23]. Agarose gel (4%) electrophoresis of the Y-branched duplexes demonstrated the formation of a single species with a mobility corresponding to 110-124 base pairs (data not shown). This is in agreement with previous reports, and suggests that the Y-shaped structure retards the migration of the duplex.[23]

Table 1. 30- and 60-base pair sequences

ID	Sequence
Fl-30	5'-GTG GCT AAC TAC GCA TTC CAC GAC CAA ATG-3'
A-30[a]	5'-CAT TTG GTC GTG GAA TGC GTA GTT AGC CAC-3'
X	5'-GCC TAG CAT GGA CTA GCG AAT TCC CGC TCT TCT CAA CTC TAG ACT CGA GGT TCC TGT CGC-3'
Y	5'-GCG TAG CCT ACG GAC TGA AGC TTA GCA GCG AGA GCG GGA ATT CGC TAG TCC ATG CTA GGC-3'
Z	5'-GCG ACA GGA ACC TCG AGT CTA GAG TTG AGA CGC TGC TAA GCT TCA GTC CGT AGG CTA CGC-3'

[a] A=acceptor; anthraquinone, rhodamine, Cy5, Cy5.5, pyrene

Fluorescence measurements were carried out using a Hitachi model F2500 fluorometer at DNA concentrations of 1.5 μM (in bases), unless otherwise specified, in 20 mM Tris-HCl buffer at either pH 7.5 for B-DNA conditions or pH 8.5 for M-DNA conditions. Fluorescein was excited at 490 nm, and the emission spectra recorded from 500-800 nm. Conversion to M-DNA was accomplished by the addition of 20 mM $ZnCl_2$ stock solution, to a final concentration of 0.2 mM,[15] except for the Stern-Volmer quenching studies where 20 mM $ZnCl_2$ was added in 2.0 μL increments. Fluorescence intensities for all samples were normalized to the fluorescein only labeled duplexes under the same conditions.

The reduction of AQ was carried out using a 0.5 mM stock solution of $NaBH_4$ (made fresh prior to reduction).[27] Briefly, the $NaBH_4$ stock solution was added to a solution of 150 μM (in bases) AQ-labeled single-stranded DNA, and incubated at room temperature for 2 hours. The reduced strand was then hybridized with the complementary fluorescein-labeled single strand to produce the fluorescein/dihydroanthraquinone labeled duplex. As a control experiment, both the fluorescein labeled single strand, as well as a fluorescein/anthraquinone duplex were also subjected to the same reduction process. Where necessary, samples were de-oxygenated by bubbling with nitrogen gas for a minimum of 30 minutes. The

reported fluorescence intensities (Tables 2 and 4) represent an average of at least 3 independent measurements, with the reported errors being the standard deviation. In order to ensure that the above procedure resulted in a reduction of the AQ group, the same procedure was carried out using 30 μM of 2-anthraquinone N-hydroxysuccinimidyl ester (AQ-NHS) in pH 8.0 10 mM Tris-HCl, 10 mM NaCl buffer. This solution was degassed by bubbling with nitrogen for 1/2 hour prior to reduction. The reduction was carried out using 0.5 M $NaBH_4$, to a final concentration of 2.5 mM. UV-vis absorbance spectra were measured before and after the reduction procedure with a Gilford 600 spectrometer. Finally, in order to determine whether or not the reduction procedure results in damage to the strands themselves, polyacrylamide gel electrophoresis (PAGE) analysis of the reduced F1-30-AQ duplexes was carried out using a 20% polyacrylamide gel.

3 Results and Discussion

3.1 Chemical Switching

The absorbance spectra of AQ-NHS (in deoxygenated buffer solution) prior to and following reduction, and upon reoxidation are shown in Figure 2.[28] Upon addition of 2.5 mM $NaBH_4$ the characteristic absorption at 335 nm disappears with a new absorption at 388 nm, which corresponds to the dihydroquinone.[27] The dihydroquinone was reoxidized to anthraquinone upon exposure to oxygen, resulting in the disappearance of the 388 nm absorption, and a reappearance of the 335 nm absorption.[29] Unfortunately, due to the high concentration of DNA required, it was not possible to carry out a similar experiment using the AQ-labeled DNA. However it is expected that the reduction will not be impacted by attachment to DNA.

The possibility of damage to the DNA strands themselves resulting from the addition of $NaBH_4$ was ruled out by gel electrophoresis studies. Figure 3 illustrates the results of PAGE analysis of both the reduced and native Fl-30-Aq duplexes; in all cases the migration of the Fl-30-Aq duplex compares well with the corresponding DNA markers. By comparing lane 5 (no added $NaBH_4$) to lanes 3 and 4 (2.5 and 25 mM $NaBH_4$, respectively) of the gel it can be seen that the reduction procedure did not result in any damage to the labeled single strand; specifically, after hybridization the untreated and treated DNA migrate to the same level. Similarly, comparing lanes 3 and 6, it can be seen that reduction of the anthraquinone labeled DNA after hybridization (lane 6) as opposed to prior to hybridization (lane 3) also did not result in any damage to the duplex itself. Finally, an ethidium bromide fluorescence assay showed binding of ethidium to the treated duplex at the same level as untreated DNA. Any damage to the duplex would have resulted in a loss of fluorescence due to decreased binding, which was not observed.

Similar to results previously reported for a number of acceptor molecules (including rhodamine,[14,15] pyrene,[15] Cy5,[15] and Cy5.5[15]) the attachment of the anthraquinone group to a fluorescein-labeled 30-mer results in significant quenching (normalized fluorescence = 0.41) of the fluorescein fluorescence upon formation of M-DNA, as shown in Table 2. Under the standard conditions used to

form M-DNA[15], namely 0.2 mM Zn^{2+} concentration at pH 8.5, the acceptor chromaphores quench the fluorescence from fluorescein by between 60-80%, while no quenching is observed at pH 7.5 (B-DNA conditions) upon the addition of Zn^{2+}.

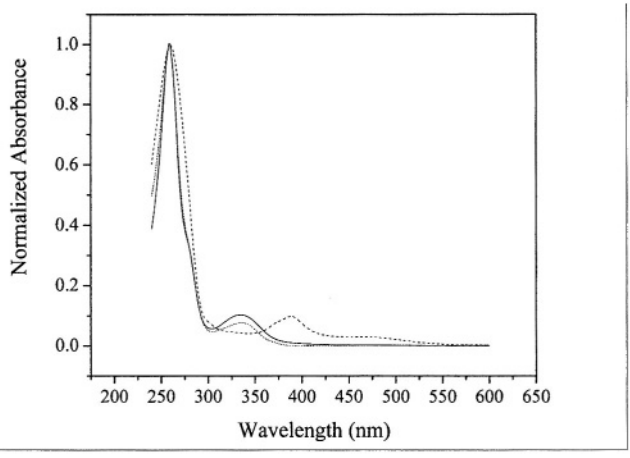


Fig. 2. Absorbance spectra of 30 μM AQ-NHS is 20 mM Tris-HCl, pH 8.5 buffer; 0 mM $NaBH_4$ (solid), 2.5 mM $NaBH_4$ (dashed), + O_2 (dotted). Adapted from reference [28]

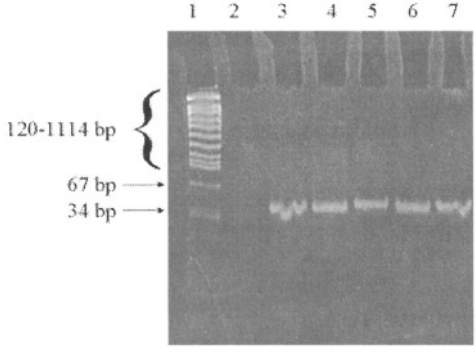


Fig. 3. Electrophorgram demonstrating the effect of anthraquinone reduction on the Fl-30-Aq duplex. Lane 1, DNA Molecular Weight Marker VIII; lane 2, empty; lanes 3 and 6, Fl-30-Aq treated with 2.5 mM $NaBH_4$; lane 4, Fl-30-Aq treated with 25 mM $NaBH_4$; lanes 5 and 7, Fl-30-Aq treated with 0 mM $NaBH_4$. For lanes 3-5, reduction carried out prior to hybridization; for lanes 6-7, reduction carried out after hybridization. From reference [28]

The results indicate that the degree of quenching is dependent upon the nature of the acceptor group and, as will be introduced below, the length of the DNA duplex. It has been suggested that the observed quenching in M-DNA may result from resonance energy transfer; however for rhodamine, Cy5 and Cy5.5 calculated efficiencies for FRET range between 0.032 and 0.053 (based upon a 30 base pair linear duplex).[15] For dyes such as anthraquinone and pyrene, the lack of spectral overlap with fluorescein indicates that resonance energy transfer is not a possible

mechanism for the de-activation of excited state fiuorescein.[30,31] Instead, the redox potentials of fluorescein as an electron donor ($E°_{Ox}$ = 0.96 V, $\Delta E_{0,0}$ = 2.46 eV) and anthraquinone as an electron acceptor ($E°_{Red}$ = -0.94 V) predict (from the Rehm-Weller equation) an exergonically favorable electron transfer process with ΔG = -0.56 eV.[28] Photo-induced electron transfer from fluorescein to anthraquinone (in molecular dyads) has previously been observed using both fluorescence quenching and ESR methods. [31,32]

Table 2. Normalized fluorescence (λ_{Em} = 520 nm) for various donor-acceptor combinations for the 30 base pair DNA duplexes; [Zn^{2+}] = 0.2 mM, pH 8.5, 20 mM Tris-HCl buffer

Duplex	Normalized Fluorescence (0 mM $NaBH_4$)	Normalized Fluorescence (2.5 mM $NaBH_4$)
Fl-30[a]	1	1
Fl-30-Rh[a]	0.14 ± 0.03	0.19 ± 0.03
Fl-30-Cy5[b]	0.18	
Fl-30-Cy5.5[b]	0.15	
Fl-30-PBA[b]	0.3	
Fl-30-AQ[a]	0.41 ± 0.04	0.71 ± 0.01
Fl-30-AQ + O_2[a]		0.42 ± 0.03

[a] from reference [28]
[b] from reference [15]

Chemical reduction of anthraquinone to dihydroanthraquinone in the M-DNA systems should result in a decrease in the fluorescence quenching of fluorescein, as it will no longer be able to accept an electron transferred from fluorescein. Figure 4 shows that this is indeed the case, with the normalized intensity from fluorescein increasing, with increasing borohydride concentration. The results obtained for the fluorescein and fluorescein/rhodamine labeled duplexes indicate that the addition of $NaBH_4$ does not result in any modification of the donor (fluorescein) group. As observed for AQ-NHS (Figure 2) the addition of oxygen results in the oxidation of the dihydroanthraquinone, and quenching is restored (see Table 2).

In order to be able to design more complicated pseudo-electronic devices from DNA, it is necessary to not only synthesize branched structures, but also to demonstrate electron transfer through the resulting junctions. The normalized fluorescence for a number of donor-acceptor combinations for the Y-branched structures is given in Table 3. It should be noted that for the fluorescein / rhodamine and fluorescein / Cy5 labeled structures the dye positions were varied among the X, Y, and Z strands and only a single result is presented; however, no dependence on dye location was observed. As observed for the linear duplexes, no quenching was observed at pH 7.5 (i.e., under B-DNA conditions), while varied levels of quenching were observed at pH 8.5 (M-DNA conditions) dependent upon the nature and number of acceptor groups. The combination of one donor with two acceptors results in the greatest amount of quenching, regardless of acceptor combination and the observed values for the normalized fluorescence compare well to that observed previously for a 54 base pair fluorescein/rhodamine labeled unbranched duplex (normalized fluorescence = 0.43[15]). This implies that the quenching mechanism, specifically

electron transfer, is not hindered by a branched junction in M-DNA. In contrast charge transfer through unstacked bases or through a branch or junction in B-DNA is either hindered [24-26], or does not occur.[33]

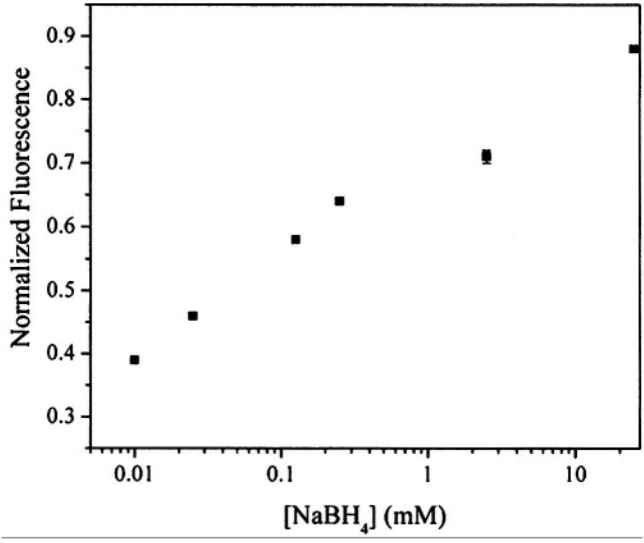


Fig. 4. Normalized fluorescence for the fluorescein/anthraquinone labeled 30-mer as a function of NaBH4 used to reduce the AQ-labeled single strand. For all measurements [DNA] = 1.5 mM; $[Zn^{2+}]$ = 0.2 mM; pH 8.5 in 20 mM Tris-HCl buffer. From reference [28]

Less quenching was observed for the case of a single acceptor (average normalized fluorescence = 0.64), compared to the results obtained for two acceptors (average normalized fluorescence = 0.41). In simplistic terms for the case of two acceptors, one could argue that there is an equal probability for electron transfer to either acceptor arm, while for a single acceptor the system behaves as the sum of two "quantum" states, one state behaving as a linear duplex with an acceptor, and the other without an acceptor. [28] This suggests that it should be possible, by appropriate choice of oligonucleotide sequence, to control the direction of electron transfer; however, the introduction of various sequence defects that could, possibly, impede electron transfer showed minimal effect. [28]

As observed above for the 30 base pair duplexes the addition of $NaBH_4$ to the fluorescein / anthraquinone labeled Y-branched duplexes again results in an increase in fluorescence emission from fluorescein, i.e., the quenching mechanism is again blocked, with the normalized fluorescence increasing from 0.77 to nearly 1. For the triple-labeled systems the addition of $NaBH_4$ results in a decrease in quenching with the normalized fluorescence increasing from 0.37 to 0.62.

By taking advantage of the switching behavior provided by the reduction of anthraquinone these systems behave in a manner analogous to that of a classical transistor, which consists of a source, a gate, and a drain. The source and drain electrodes in a transistor are separated by a semi-conducting channel, across which the potential is controlled by the gate voltage. In the Y-branched duplexes, the

fluorescein-labeled arm acts as the source, and the rhodamine-labeled arm can be thought of as the drain with the anthraquinone-labeled arm acting as the gate. The state of the anthraquinone group, i.e. reduced or unreduced, provides the means of modulating the resulting signal, in this case the emission intensity from fluorescein. Since the process is reversible by oxidation of the resulting dihydroanthraquinone each state (again using the level of fluorescence quenching as the observable) can be accessed repeatedly. As such these systems are a critical first step in the future development of more complex nano-electronic devices.

Table 3. Normalized fluorescence (λ_{Em} = 520 nm) for various donor-acceptor combinations for the Y-branched DNA junctions; [Zn^{2+}] = 0.2 mM, pH 8.5, 20 mM Tris-HCl buffer

X Strand	Y Strand	Z Strand	[$NaBH_4$] (mM)	Normalized Fluorescence
Fl		Rh	0	0.62 ± 0.01[a]
Fl		Rh	2.5	0.66 ± 0.02[a]
Fl		Cy5	0	0.70 ± 0.06
Fl		Cy5.5	0	0.68 ± 0.04
Fl	AQ		0	0.77 ± 0.01[a]
Fl	AQ		2.5	0.92 ± 0.02[a]
Fl	Rh	Rh	0	0.42 ± 0.03[a]
Fl	Rh	Cy5	0	0.36 ± 0.04
Fl	Rh	Cy5.5	0	0.34 ± 0.05
Fl	AQ	Rh	0	0.37 ± 0.01[a]
Fl	AQ	Rh	2.5	0.62 ± 0.02[a]

[a] from reference [28]

3.2 Molecular Logic

Similar to other molecular logic systems that depend on various chemical inputs,[34] M-DNA can be shown to perform a number of molecular logic functions. For example, under conditions of constant (high) pH, M-DNA behaves as a YES logic gate, with the absence (I = 0) or presence (I=1) of Zn^{2+} serving as a chemical input. The output can either be optical (i.e., fluorescence quenching) or electrochemical (i.e., impedance). For the case of donor-acceptor labeled systems introduced above, a convenient output is the level of fluorescence quenching, with the data presented in a Stern-Volmer (i.e., initial fluorescence / measured fluorescence upon addition of $ZnCl_2$, I_0/I) format. As shown in Figure 5 for a number of acceptor molecules (using fluorescein as the donor molecule) low Zn^{2+} concentrations show little quenching (I_0/I < 2)whereas above a Zn^{2+} concentration of ~0.35 mM significant quenching (I_0/I > 10) is observed. The logic table for this system, corresponding to a YES molecular logic gate, is given in Table 4.

AND logic can be demonstrated in labeled M-DNA systems by considering a second chemical input in addition to Zn^{2+}. This, for the case of any acceptor group, could be pH, where low pH (7.5) could be considered as $I_2 = 0$ and high pH (8.5)) could be considered as $I_2 = 1$. Here quenching will only be observed for conditions of

high pH in the presence of Zn^{2+}; the corresponding logic table is given in Table 5. Alternatively, the fluorescein / dihydroanthraquinone M-DNA system that results from the reduction procedure also behaves as an AND gate, considering zinc ions and oxygen as inputs. Again, quenching will only be observed for the case where both Zn^{2+} and O_2 are present.

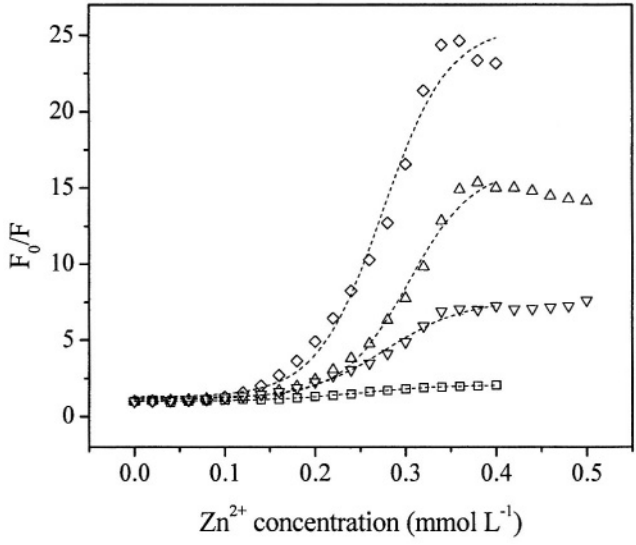


Fig. 5. Stern-Volmer plots (I_0/I, initial fluorescence/measured fluorescence upon addition of $ZnCl_2$) for various fluorescein / acceptor 30 base pair M-DNA (pH 8.5, 20 mM TRIS) systems; (□) no acceptor, (▽) anthraquinone, (△) rhodamine, (◇) Cy5

Table 4. Molecular YES logic base upon labeled M-DNA

Zn^{2+} (I)	I_0/I (O)
0 (I=0)	1 (O = 0)
0.35 (I=1)	>1 (O = 1)

Table 5. Molecular AND logic based upon the fluorescein / anthraquinone M-DNA system. I_1 = 0 or 1 corresponds to the absence or prescence of Zn^{2+}, respectively; I_2 = 0 or 1 corresponds to the absence or prescence of O_2, respectively; O = 0 or 1 corresponds to no observed quenching or observed quenching, respectively

I_1	I_2	O
0	0	0
1	0	0
0	1	0
1	1	1

Finally, the addition of a third input, in this case the absence (I_3 = 0) or presence (I_3 = 1) of EDTA, allows for INHIBIT logic to be demonstrated, again using the level of observed quenching as the observed output. The corresponding logic table is

given in Table 6. The above logic functions, combined with the demonstrated switching capabilities provided by a suitable acceptor group, in this case anthraquinone, have significant implications for the suitability of M-DNA with respect to the design of molecular nano-electronic devices.

Table 6. Molecular INHIBIT logic based upon the fluorescein / anthraquinone M-DNA system. $I_1 = 0$ or 1 corresponds to the absence or prescence of Zn^{2+}, respectively; $I_2 = 0$ or 1 corresponds to pH 7.5 or 8.5, respectively; $I_3 = 0$ or 1 corresponds to the absence or prescence of EDTA, respectively; O = 0 or 1 corresponds to no observed quenching or observed quenching, respectively

I_1	I_2	I_3	O
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0

4 Conclusions

The electron transfer process that results in the quenching of the fluorescence emission from fluorescein by anthraquinone in dye-labeled M-DNA systems is effectively blocked by chemical reduction of anthraquinone. The process is demonstrated to be reversible by oxidation of the resulting dihydroanthraquinone by exposure to oxygen. This forms the basis of a chemical switch based upon M-DNA. Contrary to previous results obtained with B-DNA branched junctions, efficient electron transfer is observed for a variety of donor-acceptor combinations in M-DNA Y-branched junctions. Combining the switching behavior observed with anthraquinone with the Y-branched structures results in a system that mimics the classical transistor. Further by considering various chemical inputs, M-DNA behaves as a number of molecular logic functions. These combined results illustrate the extraordinary potential for the application of M-DNA to the development of nano-electronic devices.

References

[1] Mao, C., Sun, W., Shen, Z., Seeman, N. C.: A nanomechanical device based on the B-Z transition of DNA. Nature 397 (1999), 144-146

[2] Yan, H., Zhang, X., Shen, Z., Seeman, N. C.: A robust DNA mechanical device controlled by hybridization topology. Nature 415 (2002), 62-65

[3] Zhang, X., Yan, H., Shen, Z., Seeman, N. C.: Paranemic cohesion of topologically-closed DNA molecules. Journal of the American Chemical Society 124 (2002), 12940-12941

[4] Niemeyer, C. M., Adler, M.: Nanomechanical devices based on DNA. Angewandte Chemie International Edition In English 41 (2002), 3779-3783
[5] Arkin, M. R., Stemp, E. D. A., Holmin, R. E., Barton, J. K., Hormann, A., Olson, E. J. C., Barbara, P. F.: Rates of DNA-mediated electron transfer between metallointercalators. Science 273 (1996), 475-479
[6] Hall, D. B., Holmlin, R. E., Branton, J. K.: Oxidative DNA damage through long-range electron transfer. Nature 382 (1996), 731-735
[7] Dandliker, P. J., Holmlin, R. E., Barton, J. K.: Oxidative thymine dimer repair in the DNA helix. Science 275 (1997), 1465-1468
[8] Porath, D., Bezryadin, A., de Vries, S., Dekker, C.: Direct measurement of electrical transport through DNA molecules. Nature 403 (2000), 635-638
[9] Rakitin, A., Aich, P., Papadopoulos, C., Kobzar, Y., Vedeneev, A. S., Lee, J. S., Xu, J. M.: Metallic conduction through engineered DNA: DNA nanoelectronic building blocks. Physical Review Letters 86 (2001), 3670-3673
[10] Storm, A. J., van Noort, J., de Vries, S., Dekker, C.: Insulating behavior for DNA molecules between nanoelectrodes at the 100 nm length scale. Applied Physics Letters 79(2001), 3881-3883
[11] Gomez-Navarro, C., Moreno-Herrero, F., de Pablo, P. J., Gomez-Herrero, J., Baro, A. M.: Contactless experiments on individual DNA molecules show no evidence for molecular wire behavior. Proceedings of the National Academy of Sciences of the United States of America 99 (2002), 8484-8487
[12] Braun, E., Eichen, Y., Sivan, U., Ben-Yoseph, G.: DNA-templated assembly and electrode attachment of a conducting silver wire. Nature 391 (1998), 775-778
[13] Richter, J.: Metallization of DNA. Physica E 16 (2003), 157-173
[14] Aich, P., Labiuk, S. L., Tari, L. W., Delbaere, L. J. T., Roesler, W. J., Falk, K. J., Steer, R. P., Lee, J. S.: *M*-DNA: A complex between divalent metal ions and DNA which behaves as a molecular Wire. Journal of Molecular Biology 294 (1999), 477-485
[15] Aich, P., Skinner, R. J. S., Wettig, S. D., Steer, R. P., Lee, J. S.: Long range molecular wire behaviour in a metal complex of DNA. Journal of Biomolecular Structure and Dynamics 20 (2002), 1-6
[16] Lee, J. S., Latimer, L. J. P., Reid, R. S.: A cooperative conformational change in duplex DNA induced by Zn^{2+} and other divalent metal ions. Biochemistry and Cell Biology 71 (1993), 162-168
[17] Gasper, S. M.: Intramolecular photoinduced electron transfer to anthraquinones linked to duplex DNA: The effect of gaps and traps on long-range radical cation migration. Journal of the American Chemical Society 119 (1997), 12762-12771
[18] Henderson, P. T., Jones, D., Hampikian, G., Kan, Y., Schuster, G. B.: Long-distance charge transport in duplex DNA: The phonon assisted polaron-like hopping mechanism. Proceedings of the National Academy of Sciences of the United States of America 96 (1999), 8353-8358
[19] Ly, D., Kan, Y., Armitage, B., Schuster, G. B.: Cleavage of DNA by irradiation of substituted anthraquinones: Intercalation promotes electron transfer and efficient raction at GG steps. Journal of the American Chemical Society 118 (1996)
[20] Armitage, B., Yu, C., Devadoss, C., Schuster, G., B.: Cationic anthraquinone derivatives as catalytic DNA photonucleases: Mechanism for DNA damage and quinone recycling. Journal of the American Chemical Society 116 (1994), 9487-9859
[21] Rajesh, C. S., Capitosti, G. J., Cramer, S. J., Modarelli, D. A.: Photoinduced Electron-Transfer within Free Base and Zinc Porphyrin Containing Poly(Amide) Dendrimers. Journal of Physical Chemistry Part B 105 (2001), 10175-10188
[22] Lilley, D. M. J.: Structures of helical junctions in nucleic acids. Quaterly Reviews of Biophysics 33 (2000), 109-159

[23] Duckett, D. R., Lilley, D. M. J.: The three-way DNA junction is a Y-shaped molecule in which there is no helix-helix stacking. EMBO Journal 9 (1990), 1659-1664

[24] Fahlman, R. P., Sen, D.: DNA conformational switches as sensitive electronic sensors of analytes. Journal of the American Chemical Society 124 (2002), 4610-4616

[25] Giese, B., Wessely, S.: The Influence of Mismatches on Long-Distance Charge Transport through DNA. Angewandte Chemie International Edition In English 39 (2000), 3490-3491

[26] Kelley, S. O., Holmlin, R. E., Stemp, E. D. A., K., B. J.: Photoinduced Electron Transfer in Ethidium-Modified DNA Duplexes: Dependence on Distance and Base Stacking. Journal of the American Chemical Society 117 (1997), 9861-9870

[27] Wightman, R. M., Cockrell, J. R., Murray, R., W., Burnett, J. N., Jones, S. B.: Protonation kinetics and mechanisms for 1,8-dihydroxyanthraquinone and anthraquinone anion radicals in dimethylformamide solvent. Journal of the American Chemical Society 98 (1976), 2562-2570

[28] Wettig, S. D., Bare, G. A., Skinner, R. J. S., Lee, J., S.: Signal Transduction Through Dye-labeled M-DNA Y-branched Junctions: Switching Modulated by Chemical Reduction of Anthraquinone. Nanoletters Articles ASAP (2003)

[29] Liu, M. D., Patterson, D. H., Jones, C. R., Leidner, C. R.: Redox and structural properties of quinone functionalized phosphatidylcholine liposomes. Journal of Physical Chemistry 95 (1991), 1858-1865

[30] Lakowicz, J. R.: Principles of fluorescence spectroscopy. 2nd ed. Kluwer Academic/Plenum Publishers, New York (1999)

[31] Zhang, H., Zhou, Y., Zhang, M., Shen, T., Li, Y., Zhu, D.: Photoinduced charge separation across colloidal TiO_2 and fluorescein derivatives. Journal of Physical Chemistry Part B 106 (2002), 9597-9603

[32] Zhang, H., Zhou, Y., Zhang, M., Shen, T., Li, Y., Zhu, D.: A Comparative Study on Photo-Induced Electron Transfer from Fluorescein to Anthraquinone and Injection into Colloidal TiO_2. Journal of Colloid and Interface Science 251 (2002), 443-446

[33] Fahlman, R. P., Sharma, R. D., Sen, D.: The charge conduction properties of DNA Holliday junctions depend critically on the identity of the tethered photooxidant. Journal of the American Chemical Society 124 (2002), 12477-12485

[34] de Silva, A. P., McClenaghan, N. D., McCoy, C. P.: Molecular logic systems. In Feringa, B. L., (ed.): Molecular Switches. Wiley-VCH Verlag GmbH, Weinheim (2001) 339-361

RCA-Based Detection Methods for Resolution Refutation

In-Hee Lee[1], Ji Yoon Park[2], Young-Gyu Chai[2], and Byoung-Tak Zhang[1]

[1] Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
[2] Department of Biochemistry and Molecular Biology
Hanyang University, Ansan, Kyongki-do 425-791, Korea
{ihlee,jypark,ygchai,btzhang}@bi.snu.ac.kr

Abstract. In molecular resolution refutation, the detection of empty clauses is important. We propose a rolling circle amplification-based detection method for resolution refutation. The rolling circle amplification (RCA) technique is known to be able to distinguish and amplify circular DNA. In this paper, we describe the representation of clauses and the RCA-based detection method. Bio-lab experiments show the basic idea for this method is correct.

1 Introduction

Many researchers utilized the massive parallel reaction of DNA molecules to perform logical computation [8, 9]. In our previous work, we proposed a DNA computing method for theorem proving with resolution refutation in propositional logic using hybridization, PCR and gel electrophoresis [5]. But, as indicated in that paper, there was some difficulty with the PCR step because of DNA structure.

In this paper, we propose a detection method using the rolling circle amplification (RCA) technique to overcome these difficulties. RCA is known to be able to amplify circular DNA, and can be used to distinguish various topological structures [4]. As will be explained later, the empty clause has circular form. Therefore, RCA can be used to detect and amplify the empty clause.

In the following, the improved method of molecular resolution refutation will be described. We performed a simple experiment to test the feasibility of the proposed method and provided the result. In addition, we applied the method for proving the pigeon hole principle in lab experiment (in progress).

The rest of the paper is organized as follows. A brief introduction to the pigeon hole principle (PHP) and RCA is given in Section 2. Section 3 will describe experimental results and conclusions are drawn in Section 4.

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 32–36, 2004.

2 Methods

2.1 Resolution Refutation and the Pigeon Hole Principle

Refutation is a technique which proves the target formula by showing that the negation of the goal results in inconsistency. Resolution refutation is a kind of refutation which uses the resolution when showing the inconsistency. It requires that every formula is expressed in clause form. A clause form in propositional logic is defined as a set of literals connected with disjunctions (V). A clause which contains no literal is called an empty clause *(nil).* From two clauses $\mathcal{A}$ and $\mathcal{B}$, resolution draws $(\mathcal{A} - \{v\}) \vee (\mathcal{B} - \{\neg v\})$, where v be a literal such that $v \in \mathcal{A}$ and $\neg v \in \mathcal{B}$. We say that we resolved $\mathcal{A}$ and $\mathcal{B}$ on v and the product of resolution is called a *resolvent.* During applying resolution repeatedly, if a *nil* is produced, it is shown that the given clauses are not consistent. For it is proven that the negation of the goal leads to inconsistency, the goal is proven.

As a benchmark problem for the scaled-up version of our previous work [5], we choose the PHP which has been used as a test case for a proof system. Generally speaking, the PHP means a tautology which states that there exists no one-to-one mapping from m objects (pigeons) to n objects (holes) where $m > n$, and is denoted as PHP_n^m [1]. To prove PHP_n^m with resolution refutation, we need to express the negation of PHP_n^m in clause form as follows:

1. $P_{i,1} \vee P_{i,2} \vee \cdots \vee P_{i,n}$, for each $i \leq m$, and
2. $\neg P_{i,k} \vee \neg P_{j,k}$, for each $i, j \leq m, k \leq n, i \neq j$,

where $P_{i,j}$ denotes the mapping of pigeon i to hole j. In this paper, PHP_2^3 will be considered. The clauses of $\neg PHP_2^3$ and one of its proof tree are given in Fig. 1-(a,b).

2.2 RCA-Based Detection of Empty Clauses

Before explaining the RCA-based detection method for the empty clause, let us explain molecular representation of clause form. As in [8], each variable is encoded with a unique sequence. And the negation of the variable is represented with its complementary sequence. A clause with c literals is represented with molecules of c branches. Each branch has a sticky end whose single-stranded region corresponds to each literal in the clause.

With this scheme, the hybridization of complementary sticky ends means the resolution of a literal from the clauses. To prevent inverse resolution, we ligated the hybridization mixture. In our example of PHP_2^3, an empty clause will have the form of a double-stranded ring as illustrated in Fig. 1-(c). In this way, each DNA ring represents a proof tree.

Now, all that is left is the detection of the empty clause, if any. Detecting a small amount of DNA is very difficult. Therefore, we used RCA to amplify the empty clause selectively, for the empty clause has a circular structure. To get more clear output, we deleted the non-empty clauses using exonuclease-III. Since the non-empty clauses have at least one sticky end, exonuclease-III can digest them.

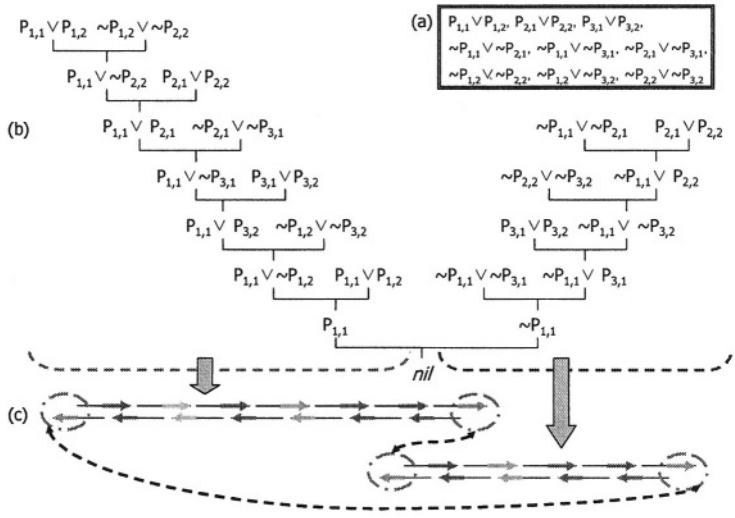


Fig. 1. (a) The clauses of $\neg PHP_2^3$. (b) One of possible resolution proof trees. (c) The molecular proof tree constructed from (b). The pair of ellipses connected to each other by dashed arrows represent complement sticky ends. If these two sticky ends hybridize together, a double-stranded DNA ring will be constructed

3 Experimental Results

We tested the principle of the RCA-based method on a test problem. We test whether the RCA technique can be used to distinguish circular DNA from others. More specifically, we tried to amplify molecules of three different kinds of structure. The three kinds of structure compared are sticky ends, blunt ends, and a circular form. If only circular DNAs are amplified, we can conclude that the empty clause can be detected and amplified by RCA.

The sequences used to make sticky ends and blunt ends were designed by NACST/Seq using evolutionary optimization as described in [7]. For a circular DNA, we used the pUC19 plasmid. All oligonucleotides were synthesized by Genotech Co. The designed sequences are shown in Table 1. To determine the condition for the enzyme reaction, we referenced [2, 3, 6]. The experiment includes hybridization, reaction with exonuclease-III, RCA and gel electrophoresis.

Table 1. Sequences designed for the test experiment

Sequences with two sticky ends	
3'-	CGTACGTACGCTGAACTGCCTTGCGTTGACTGCGTTCATTGTATG-5'
5'-TTCGTCATCGCTGGTTAACTGCGTTTAACTGCATGCATGCGACTT	-3'
Sequences with blunt ends	
3'-AAGCAGTAGCGACCAATTGACGCAAATTGACGTACGTACGCTGAACTGCCTTGCGTTGACTGCGTTCATTGTATG-5'	
5'-TTCGTCATCGCTGGTTAACTGCGTTTAACTGCATGCATGCGACTTGACGGAACGCAACTGACGCAAGTAACATAC-3'	

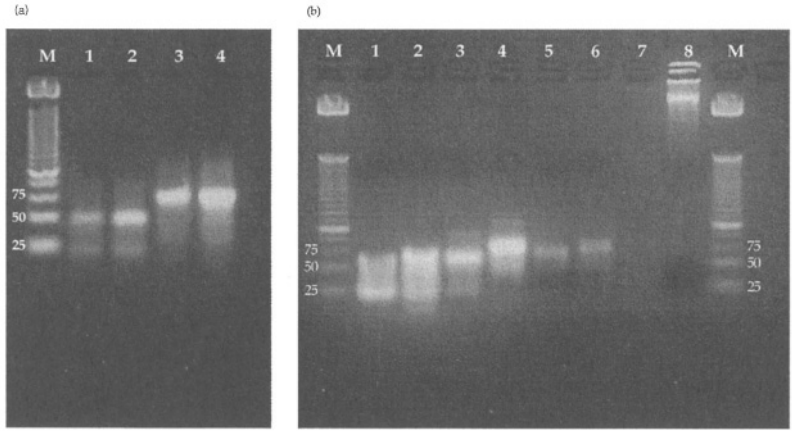


Fig. 2. (a) The electrophoresis of oligonucleotide mixture of blunt ends and sticky ends on 3% agarose gel. Lanes 1,2: hybridization result of sticky ends. 100 pmol and 200 pmol of the mixture, respectively. Lanes 3,4: hybridization result of blunt ends. 100 pmol and 200 pmol of the mixture, respectively. Lane M: 25 bp ladder, (b) The electrophoresis of RCA results on 3% agarose gel. Lanes 3,4: hybridization result of blunt ends and sticky ends respectively. Lanes 5,6: RCA product of lanes 3 and 4, respectively. Lane 7: Exonuclease III digested product of lanes 3 and 4. Lane 8: RCA product of pUC19. Lane M: 25 bp ladder

Fig. 3 shows the result of hybridization and gel electrophoresis. To confirm the formation of sticky ends and blunt ends, we analyzed the hybridization mixture by gel electrophoresis as shown in Fig. 3 (a). As can be seen in lane 5 of Fig. 3 (b), the molecules with blunt ends or sticky ends were amplified by RCA. But in comparison to lane 8, the positions of bands in each lane are different. Amplifying circular template with RCA produces very long strands. On the contrary, RCA with linear template produces strands of the same length with the original. As a result, with the RCA method, we can distinguish circular DNAs from the others.

The results demonstrate that RCA can amplify the circular DNAs selectively, and thus can be used to detect the empty clauses. Based on this result, we applied the method to proving PHP_2^3. In this case, the overall procedure is the same as that of the previous experiment. But, the ligation step is included after hybridization. In this ligation step, all clauses in a proof tree get connected and are never separated in the following steps. The experiment for this problem is in progress.

4 Conclusions

We presented an RCA-based detection method for solving theorem proving problems. By trying to amplify molecules with various structures, we showed that

RCA can be used to detect a circular DNA. Since the empty clause has a circular form in our representation, this result supports the RCA-based detection method for empty clauses. Because our method uses a constant number of simple lab operations, it is highly extensible. Also, with the help of lab-on-a-chip technique, it may be automated.

Acknowledgements

This research was supported in part by the Ministry of Education & Human Resources Development under the BK21-IT Program, the Ministry of Commerce, Industry and Energy through MEC project, and the NRL Program from Korean Ministry of Science and Technology. The RIACT at Seoul National University provides research facilities for this study.

References

[1] Buss, S. R. and Pitassi, T., Resolution and the Weak Pigeonhole Principle, *Lecture Notes in Computer Science vol. 1414,* 149–156, 1998.

[2] Dean F. B., Nelson, J. R., Giesler, T.L., and Lasken, R. S., Rapid Amplificaion of Plasmid and Phage DNA using phi29 DNA Polymerase and Multiply-primed Rolling Circle Amplification. *Genome Research,* **11**:1095-1099, 2001.

[3] Henikoff, S. Unidirectional Digestion with Exonuclease III Creates Targeted Breakpoints for DNA Sequencing. *Gene,* **28**:351-359, 1984.

[4] Kuhn, H., Demidov, V. V., and Prank-Kamenetskii, M. D., Rolling-circle Amplification under Topological Constraints, *Nucleic Acid Research,* **30**(2):574–580, 2002.

[5] Lee, I.-H., Park, J.-Y., Jang, H.-M, Chai, Y.-G, and Zhang, B.-T., DNA Implementation of Theorem Proving with Resolution Refutation in Propositional Logic, *Proceedings of the Eighth International Meeting on DNA Based Computers,* 251–260, 2002.

[6] Rogers, S.G. and Weiss, B., Exonuclease III of Escherichia Coli K-12, an AP Endonuclease. *Methods in Enzymology,* **65**:201-211, 1980.

[7] Shin, S.-Y., Kim, D., Lee, I.-H., and Zhang, B.-T., Evolutionary Sequence Generation for Reliable DNA Computing, *Proceedings of 2002 IEEE World Congress on Evolutionary Computation,* 79–84, 2002.

[8] Uejima, H., Hagiya, M., and Kobayashi, S., Horn Clause Computation by Self-assembly of DNA Molecules, *Proceedings of the Seventh International Meeting on DNA Based Computers,* 308–320, 2001.

[9] Wasiewicz, P., Janczak, T., Mulawka, J. J., and Plucienniczak, A., The Inference based on Molecular Computing, *International Journal of Cybernetics and Systems,* **31**(3):283–315, 2000.

Word Design for Molecular Computing: A Survey

G. Mauri and C. Ferretti

Dipartimento di Informatica Sistemistica e Comunicazione
via Bicocca degli Arcimboldi 8, 20136 Milano
Università degli Studi di Milano-Bicocca - Italy

Abstract. This paper gives a short survey, and a reading grid, of results about the coding problem of molecular design.
This work has been funded by European MolCoNet Project.

1 The Problem of Designing Molecules

Molecular computing exploits biological reactions naturally occurring in the test tube (or *in vivo*). The goal of an experiment is a given set of final molecules, we have the set of reactions, and thus we must design a set of starting molecules that, going through those reactions, produce the desired result.

A simple instance of these steps could be one where we want to produce a long DNA sequence, starting from designed shorter ones and by exploiting the reaction of nucleotide complementarity and hybridization. We obviously can't start from arbitrary short sequences, we are instead required to design them so that the rules, defined by the reaction, will allow their spontaneous assembly into the longer one.

We could call this a *positive* design problem: we design a set of molecules, formally considered as words, such that there exists *a way* for the reaction to correctly assembly them into the desired longer molecule.

The trouble is that the reaction will also allow those short molecules to assembly in other different and undesired ways, thus damaging the correctness (we have wrong molecules) or at least the efficiency (we have correct final molecules mixed with wrong molecules) of the experiment. For instance, we could have designed two single stranded DNA molecules `5'-AACCAAGG-3'` and `3'-TTCCAACC-5'`: they would build the longer, partially hybridized, molecule

```
5'-AACCAAGG-3'
      ||||
  3'-TTCCAACC-5'
```

but they could also produce the undesired molecule built by sticking together the two leading bases and the two trailing bases, which also are complementary.

Then we have to introduce a *negative* design problem: we design a set of molecules that (also) gives no way to the reaction to produce undesired molecules.

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 37–47, 2004.

Lower Level Details. Keeping the focus on the instance reaction of hybridization, we could go to a lower level of its physical details, moving from the rules just stating the complementarity of the bases, down to the equations giving measures of how strong could be the hybridization of two given complementary single strands. In fact, if we take two double stranded DNA molecules, having the same length, the strength of their hybridizations could be different, due to the different bases they are made with.

This has interesting effects on the efficiency of molecular biology experiments, and thus also for molecular computing. When designing the molecules to be used in an experiment, e.g. where we would use hybridization, we could try to solve the afore mentioned positive and negative design problem while also having the further goals of

- keeping the strength of correct hybridizations *higher* than that of wrong ones,
- keeping the strength of all correct hybridizations reasonably *uniform* in value, so that all of them would eventually show up in the test tube with similar probability, and so that all of them would evenly suffer from the usual experimental errors due to temperature and concentration instabilities.

Technically speaking, the measures we are considering are those of *free energy* and *melting temperature.* Their values range in continuous domains, depending on the exact sequence of bases building the molecules being measured. If we'd move to reactions different from hybridizations, we would be subjected to other continuous physical measures affecting, in the experiments, the efficiency of our designed molecules.

2 Research Avenues in Word Design

If we keep the two categories of positive and negative design problems, we can start analyzing the results in literature by observing that many early papers approached the first kind of problem. In this case the resulting design is usually strongly related to the specific requirements of a given proposed experiment, and it is not easy to be generally categorized, or ported to different experiments.

The negative design problem, instead, can be stated in a more general way: we look for a *library* of sequences which are known to avoid (undesired) transformations when processed together in a given type of reaction. A solution to this formulation of the negative problem would allow to solve some positive problem by fetching from such a library a set of sequences useful for a specific proposed experiment.

A different way to organize the results in literature, and in the sampled papers we'll shortly review, can be to group them along the following categories of results:

1. theoretical models, to study general properties of libraries
2. theoretical models, to estimate bounds on the size of a library
3. algorithms to design the libraries

4. software tools

The first two categories of results usually try to map techniques derived from the classical theory of codes to the requirements imposed by molecular reactions on words. For instance, two words in a library should be far enough one from the other, but while in usual codes by "far" we would mean something like Hamming distance, when considering DNA molecules we need to define new types of distance: two complementary molecules would be far in usual codes, but should be at null distance in a formal model of DNA codes.

Third and forth categories are often based on algorithms which do not look for perfect or optimal solutions, since the space that should be explored would be too large. This leads to many results suggesting which quality measure gives the best compromise between physical accuracy and required computational time. Alternatively, many researchers use probabilistic algorithms, such as those of the genetic programming field.

Theoretical results help the study of algorithms and software by giving either constructive proofs of existence of optimal libraries, even if this kind of result is still much rarer than what is available in classical coding theory, or by giving estimates of the maximal sizes of libraries, so that one could know how much an existing algorithm could be ideally improved.

Moreover, the results in each of these categories could take into account the lower level details of the reactions, even if this means to move from the basic design, which is a discrete combinatorial problem, to a problem where the lower details introduce continuous domains.

3 Theoretical Models

Theoretical models related to the coding problem can be derived from classical theory of codes (e.g., error correcting codes) but, for instance, Watson-Crick complementarity is a new feature to be taken into account and formalized.

We introduce some notation: Σgeneric alphabet, $\Delta = \{A,C,G,T\}$,
$\Sigma^* =$ set of words over Σ, 1 = empty word, $\Sigma^+ = \Sigma \setminus \{1\}$;
w^{RC} = Watson-Crick reverse-complement of $w \in \Delta^*$, $|w| =$ length of w.
Code $:= C \subseteq \Sigma^*$; DNA code $:= C \subseteq \Delta^n$ (equilength words)
C uniquely decipherable $:= \forall w \in C^* : \exists! w_1, w_2, ..., w_n (w = w_1 w_2 ... w_n)$

We give the definition of Hamming distance and of some of its variants:
$H(x,y) := |\{k : x_i \neq y_i\}|$ (Hamming distance H)
$H^R(x,y) := |\{k : x_i \neq y_i^R\}|$ (reverse H)
$H^{RC}(x,y) := |\{k : x_i \neq y_i^{RC}\}|$ (reverse complement H)
$Hs(x,y) := \min_{-n<k<n} H(x, \sigma^k(y^{RC}))$ (H^{RC} with frame shift), where $\sigma^k = k$ positions shift (left or right) [D+96, G+97, MCC01]
$H_M(x,y) := \min\{H(x,y') \mid y'$ subword of y, $|y'|=n\}$ (for $|y|>|x|$) [KKA02]

Imposing Constraints

Theoretical results can be grouped according to the imposed constraints (see for instance [MCC01, KKA02]). We consider four of such constraints, or properties, and we label them with letters A through D. Please note that these labels will also be used again in this paper when summarizing the properties of the coding algorithms we survey.

- Three combinatorial constraints

These are defined in terms of variants of Hamming distance:

A) $H(x, y) \geq d$
B) $H^{RC}(x, y) \geq d$
C) $H_M(x, yz) \geq d$, and any combination of R, C variants

- Thermodynamical constraints

These are related to the low level physical details, and we consider only the following property:

D) words must have similar GC content, or similar melting temperature (Breslauer estimate of free energy can be used)

Each of these constraints are motivated by experimental requirements, in details:

A) Similarity between words must be limited to ease correct detection by hybridization

B) We must avoid having a word similar to the reverse complement of another word because they could hybridize in the 3D space of the test tube. Observation ([MCC01]): construction of codes satisfying $x, y \in C$, $H(x, y^R) > d$ can easily suggest how to construct codes satisfying property B.

C) Similarity could arise between a (reverse complement of a) codeword and the concatenation of two codewords in test tube.

D) Having the test tube at a given temperature should have the same effect, in terms of melting or hybridization, on all words in it. See for instance [R+97].

Results in Literature

The papers [KKT00] and [HKK01] deal quite extensively with "Good properties" of DNA codes as a particular case of general algebraic properties of codes wrt morphic (antimorphic) involutions. We summarize here those papers.

Definition. Involution : $= \theta$: $\mathrm{R} \rightarrow \mathrm{R}$ with $\theta^2 = \mathrm{I}$. This definition establishes a link to the behaviour of DNA molecules, since the mirror (reverse) μ , the complement γ and the Watson-Crick reverse complement $\mu\gamma = \gamma\mu = \tau$ transformations on Δ^* are involutions. The focus of the paper is on further properties defined on languages and codes as follows:

Definition. $C \subseteq \Sigma^*$ θ-compliant $:= \forall x, y, u \in \Sigma^*$: $u, x\theta(u)y \in C \Rightarrow xy = 1$ (*)

$C \subseteq \Sigma^*$ strictly θ-compliant := above property (*) and $C \cap \theta(C) = \emptyset$ hold

Definition. $C \subseteq \Sigma^*$ prefix θ-compliant := $\forall y, u \in \Sigma^*$: $u, \theta(u)y \in C \Rightarrow y = 1$ (*)

$C \subseteq \Sigma^*$strictly prefix θ-compliant := above property (*) and $C \cap \theta(C) = \emptyset$ hold

Definition. $C \subseteq \Sigma^*$ suffix θ-compliant := $\forall x, u \in \Sigma^*$: $u, x\theta(u) \in C \Rightarrow x = 1$ (*)

$C \subseteq \Sigma^*$ strictly suffix θ-compliant := above property (*) and $C \cap \theta(\mathrm{C}) = \emptyset$ hold

Definition. $C \subseteq \Sigma^*$ θ-free := $C^2 \cap \Sigma^+\theta(C)\Sigma^+ = \emptyset$ (*)

$C \subseteq \Sigma^*$ strictly θ-free := above property (*) and $C \cap \theta(C) = \emptyset$ hold

Proposition. C strictly θ-free $\Rightarrow C^+$ strictly θ-free

These definitions mirror some desired properties of DNA sequences in an experiment for molecular computing since, for instance, for $\Sigma = \Delta$ and $\theta = \tau$, being $C \subseteq \Delta^*$ strictly θ-free means that C is a DNA code with avoids hybridization among the concatenation of any two words and the Watson-Crick complementarity of another word.

The results in the paper give some characterizations of these properties and some methods to obtain languages enjoying them. Finally, splicing systems preserving those properties are studied.

Another interesting theoretical paper, that we summarize here, is by Head [H02] and deals with properties of codes which can be relativised to specific subsets of strings, with consequences for languages built on DNA base pairings. The chosen formal setting includes the following definitions:

Definition. $C \subseteq \Sigma^*$ solid :=

1. $\forall x, y, u \in \Sigma^*$: $u, xuy \in C \Rightarrow xy = 1$ (i.e., I-compliant) and
2. $\forall x, y, u \in \Sigma^*$: $xu, uy \in C$ and $u \neq 1 \Rightarrow xy = 1$ (a prefix of a codeword cannot be suffix of another one).

$C \subseteq \Sigma^*$ solid relative to $L \subseteq \Sigma^*$:= (1) and (2) hold only for $w = pxuyq \in L$

Definition. $C \subseteq \Sigma^*$ comma-free:= C solid relative to C^*

Definition. $w \in \Sigma^*$ join relative to $L \subseteq \Sigma^*$:= $\exists xwy \in L$ and $(xwy \in L \Rightarrow x, y \in L)$

$w \in C$ join in C := w join relative to C^*; $J(C)$:= set of joins in C

Definition. Let C uniquely decif., $C_{0=C}$, $C_{i+1} = C_i \backslash J(C_i)$ $(J(C_i) \neq \emptyset)$

C join code of level k := $C_k = \emptyset$ and $C \neq \emptyset$ for $i < k$

C join code of infinite level := $C \backslash \bigcup_i J(C_i) = \emptyset$

Among the results of this paper we quote:

Theorem. Relative solidity is decidable for L, C regular

Theorem. Comma-freeness is decidable for C regular

Moreover, join codes of level 1 are the comma-free codes, and the concept of join codes of level k extends that of comma-freeness in the sense that even for them an easy parsing of a long message, or sequence, can be accomplished, as a sequence of k parsing steps.

An application of these result is then suggested: decoding of ssDNA messages based on a finite join code of level k can be carried out with k washes.

4 Bounds on the Size of DNA Codes

We consider in this section the theoretical results which specifically try to evaluate the maximum size of codes verifying some given constraints.

One of the first results ([Ba96]) gives the maximum size of a code C s.t. if SvS=uxt and (SwS=u'xt' or SwS=u'x^{RC}t') then —x—¡k ($v, w \in C$; S spacer) The bound in this case is $3 * 4^{k-2} + 5 * 4^{k-4} - 2 * 4^{k/2-1}$.

The paper [G+97] numerically explores the maximal size of codes with: $Hs(x, y) \geq t$, —x—=$n \leq 9$, –Σ—=4.

The paper [MCC01] defines and studies the following size measures:
$A_q^{RC}(n, d)$:= max size of a RC code with constraint d
$A_q^{R}(n, d)$:= max size of a R code with constraint d
$A_q(n, d)$:= max size of a code with constraint d
(with n word's length, d min mismatches, q alphabet size)
A few results are then obtained as a kind of "porting" of classical results in coding theory to the new setting: the paper presents bounds on $A_4(n, d)$, $A_2^R(n, d)$, $A_4^R(n, d)$.

5 Algorithms

We present now the main features of some papers giving interesting results concerning algorithms. For each algorithm, we put in evidence which of the properties or constraints A) through D), previously defined, are more closely taken into account by the algorithm itself. (We do not focus on the problem of designing molecules to build complex spatial structures, which is studied, for instance, by [BC01, HCH02].)

– Baum [Ba96]:

Constraints: AB– (no long common substring)
Structure: Input: k
Choose b∈ Δ and generate the spacer b^k; codewords will have the form b^kxb^k, with —x—=k
The (variable) inner sequences are then chosen with care from classes of (partially) complementary words; b and bC have to be avoided at the ends of inner sequences
Maximal codes are generated.

– Memphis MolComp Group [D+96] (and [D+96b]): Constraints : –BC-

Genetic algorithm: Input: —C—, d, n (allowed mismatches for hybridization)
Hamming reverse-complement distance considered
It evolves codes with words of length 20 that avoid 5 types of undesired word matches
Tailored on Adleman's solution of the Hamiltonian Path Problem.

– Memphis MolComp Group [G+97]: Constraints : -BC-

Greedy sieve algorithm: Input: n, —C—, d
select codewords with good Hamming reverse complement distance (with shift)
An upper bound, exponential in the length of codewords, is thus obtained on the time required to build such a code: $O(—C—^{4n})$.

– Frutos et al. [F+97]: Constraints: ABCD

The paper shows how to build DNA words carrying data (4-8 bits) in sequences terminating with fixed word labels
The algorithm starts from a set of templates and then extends it
Thermodynamic stability is estimated by a simple method (pairwise additive).

– Reif, LaBean [RLb00]: Constraints : A— (consider solid support on chips)

They apply known methods from error-correcting codes and vector-quantization to select good encodings. These encodings are suggested for DNA chips, transformation of data toward/from DNA, and the process of affinity separation.

– Marathe, Condon, Corn [MCC01]: Constraints : AB-D

AB are satisfied by extending classical code construction methods, and by using greedy procedures to eliminate codewords violating constraints
for D: dynamic programming algorithm: Input: n, ΔG
Compute the number of words of length n with free energy ΔG (Breslauer approx formula)
Randomly generate a word. Time: $O(n^2)$.

– Tulpan, Hoos, Condon [THC02]: Constraints : AB-D

Stochastic local search: Input: n + —C— + constraints (Hd, RC, GC)
C := random choice
Select x, y ∈ C that violate the constraints. Correct x or y
Stop when no violations or after k iterations
Occasional random replacement of a few words
Experimental results are reported in the paper.

– Kobayashi, Kondo, Arita [AK02, KKA02]: Constraints : ABCD

It uses a set of templates for codewords: Input: n, d
Generates C satisfying constraint d in three steps:
a binary template for the GC content is given (exhaustive search for $n \leq 30$)
a binary error correcting code is chosen, satisfying d constraint
DNA words are generated by masking the template with each (binary) codeword.

- Ben-Dor et al. [BKSY00]: Constraints : A–D (on DNA chips)

They start from cyclic De Brujin sequences, and then a greedy algorithm select codewords (tags) on them. A paper on a similar problem, but with with an approach closer to that of molecular computing is [RDHS01].

- Kaderali, Schliep [KS02]:

The paper presents an interesting way to design of codes with good melting temperatures by using suffix trees.

6 Software

Many papers make use of some in-house-developed software tools, see for instance [KS02] or the couple of complementary papers [D+02, D+02b]. Some papers focus on the software itself. An interesting questions could be: do they just implement algorithms presented by other papers? Usually they don't.

We can list a couple of known "pre-molcomp" software, both developed to design sequence able to build some desired spatial structures:

- DeNovo [S90]
- Vienna package (RNA folding) [H+94]

More recent packages have been developed specifically for molecular computing:

- SCAN [HGK98]: the software accepts a set of constraints, derived from the design of an experiment, and it looks for encoding satisfying them and having good melting temperature. Performance results and comparisons to manually selected sequences are given.
- DNASequence Generator [FBR00, FSBR01]: this software tool is able to build libraries of DNA sequences which comply with a required set of logical and low level physical properties. Moreover, the library can be built so that it includes and then extends a set of manually given sequences.
- NACST/Seq [KSLZ02]: this is a DNA sequence optimization system, based on genetic algorithms. It integrates a DNA sequence analyzer able to visualize some properties of given sequences.

7 Observations

We can see from the cited papers that the field of molecular word design is very active, and actually crossing many different fields.

We found interesting to answer some questions by making cross checks between papers of various type, e.g.: Do experimental papers use published algorithms and/or software? Do publicly available software programs implement previously published algorithms? Do published constructive theoretical results have complete algorithms and/or software implementations?

Unfortunately, in this research area it's difficult to shortly describe open problems, since the specific goals of each involved field can be so different. Nonetheless, some emerging theoretical open problems can be quickly mentioned: to complete the theory of codes with reverse complement; Reverse-complement stringology; Counting: bounds on size of codes over a 4-letter alphabet properties of shift-adjusted distance; new metrics: local constraints (involving codewords) vs global constraints (e.g., secondary structure freeness); Algorithms Comparison of different heuristic methods.

It would be of great value to keep researches on the coding problem more autonomous with respect to the various fields applying them. Those researches would be coordinated so to produce general results which could be adopted and applied by the other fields. Many of the papers available in literature seem, for the time being, to originate from several small families of researcher groups, and it's a little hard to apply, or accumulate, the results obtained by a family to the work of another "research family". Subtle variations in the definitions of properties A)-D) are obstacles to the integration of results Algorithms which generate codes could be labeled with respect to the kind of experiment they could feed, e.g.:

- Self assembly (properties such as C)
- Solid support (and DNA chip) (property A only)
- Associative memory (property B)

A final observation is that it's worth to study a way to produce tools, in terms of software and laboratory protocols, which are at the same time general and helpful to all the different experimental needs, strongly felt by so many groups. It would be precious to have some kind of tool available "off the shelf", or at least some way too share and coordinate the production of software for the molecular design, in a way similar to what it's happening for software products approaching the complementary problem of sequence (DNA, ...) analysis.

And therefore, what about unifying software architectures ? When developing software, a labeling similar to that suggested for algorithms could be applied. A supporting architecture could be developed, where each software tool could be inserted as a plug-in, centralizing GUI and promoting interchangeability of data (code libraries, etcetera).

References

[AK02] M. Arita, S. Kobayashi, "DNA sequence design using templates", New Generation Computing, 20: 263-, 2002.

[Ba96] E. B. Baum, "DNA sequences useful for computation", Proc. 2nd DIMACS Workshop on DNA Based Computers, June 1996.

[BKSY00] A. Ben-Dor, R. Karp, B. Schwikowski, Z. Yakhini, "Universal DNA tag systems: a combinatorial design scheme", J. Computational Biology, 7:3/4, 2000.

[BC01] A. Brenneman, A. E. Condon, "Strand design for bio-molecular computation", Theoretical Computer Science, 287(1): 39-, 2002.

[D+02] R. Deaton, J. Chen, H. Bi, M. Garzon, H. Rubin, D.H. Wood, "A PCR-based protocol for *in vitro* selection of non-crosshybridizing oligonucleotides", Proc. 8th DIMACS Workshop on DNA Based Computers, June 2002.

[D+02b] R. Deaton, J. Chen, H. Bi, J. Rose, "A software tool for generating non-crosshybridizing libraries of DNA oligonucleotides", Proc. 8th DIMACS Workshop on DNA Based Computers, June 2002.

[D+96b] R. Deaton, M. Garzon, R. C. Murphy, J. A. Rose, D. R. Franceschetti, S. E. Stevens, Jr., "Genetic search of reliable encodings for DNA-based computation", in Late breaking papers at the Genetic Programming conference, Stanford University, July 1996.

[D+96] R. Deaton, R. C. Murphy, M. Garzon, D.R. Franceschetti, S.E. Stevens, Jr., "Good encodings for DNA-based solutions to combinatorial problems", Proc. 2nd DIMACS Workshop on DNA Based Computers, June 1996.

[FBR00] U. Feldkamp, W. Banzhaf, H. Rahue, "A DNA sequence compiler", Proc. 6th International Meeting on DNA Based Computers, 2000.

[FSBR01] U. Feldkamp, S. Saghafi, W. Banzhaf, H. Rahue, "DNASequenceGenerator: a program for the construction of DNA sequences", Proc. 7th International Meeting on DNA Based Computers, 2001.

[F+97] A. G. Frutos, Q. Liu, A. J. Thiel, A. W. Sanner, A. E. Condon, L. M. Smith, R. M. Corn, "Demonstration of a word strategy for DNA computing on surfaces", Nucleic Acid Research, vol.25, issue 23, 1997.

[G+97] M. Garzon, R. Deaton, P. Neathery, R. C. Murphy, D. R. Franceschetti, S.E. Stevens, Jr., "On the encoding problem for DNA computing", Proc. 3rd DIMACS Workshop on DNA Based Computers, 1997.

[HGK98] A. J. Hartemik, D.K. Gifford, J. Khodor, "Automated constraint-based nucleotide sequence selection for DNA computation", Proc. 4th DIMACS Workshop on DNA Based Computers, June 1998.

[H02] T. Head, "Relativised code concepts and multi-tube DNA dictionaries", submitted, 2002.

[HCH02] C. E. Heitsch, A. E. Condon, H. H. Hoos, "From RNA secondary structure to coding theory: a combinatorial approach", Proc. 8th DIMACS Workshop on DNA Based Computers, June 2002.

[H+94] I. L. Hofacker, W. Fontana, P. F. Stadler, S. Bonhoffer, M. Tacker, P. Schuster, "Fast folding and comparison of RNA secondary structures", Montashefte f. Chemie, vol.125, 1994.

[HKK01] S. Hussini, L. Kari, S. Konstantinidis, "Coding properties of DNA languages", Theoretical Computer Science, 290: 1557-, 2003.

[KS02] L. Kaderali, A. Schliep, "An algorithm to select target specific probes for DNA chips", to be published by Bioinformatics, 2002.

[KKTOO] L. Kari, R. Kitto, G. Thierrin, "Codes, involutions and DNA encoding", Formal and Natural Lecture Notes in Computer Science 2300, 376-, 2002.

[KSLZ02] D. Kim, S-Y. Shin, I-H. Lee, B-T. Zhang, "NACST/Seq: a sequence design system with multiobjective optimization", Proc. 8th DIMACS Workshop on DNA Based Computers, June 2002.

[KKA02] S. Kobayashi, T. Kondo, M. Arita, "On template method for DNA sequence design", Proc. 8th DIMACS Workshop on DNA Based Computers, June 2002.

[MCC01] A. Marathe, A. E. Condon, R. M. Corn, "On combinatorial DNA word design", J. Computational Biology, 8:3, 2001.

[RLb00] J. H. Reif, T. H. LaBean, "Computationally inspired biotechnologies: improved DNA synthesis and associative search using error-correcting codes and vector-quantization", Proc. 6th DIMACS Workshop on DNA Based Computers, June 2000.

[R+97] J. A. Rose, R. Deaton, M. Garzon, R. C. Murphy, D. R. Franceschetti, S. E. Stevens, Jr., "The effect of uniform melting temperatures on the efficiency of DNA computing", Proc. 3rd DIMACS Workshop on DNA Based Computers, 1997.

[RDHS01] J. A. Rose, R. J. Deaton, M. Hagiya, A. Suyama, "The fidelity of the Tag-Antitag system", Proc. 7th International Meeting on DNA Based Computers, 2001.

[S90] N. C. Seeman, "De novo design of sequences for nucleic acid structural engineering", J. Biomolecular Structure and Dynamics, 8:3, 1990.

[THC02] D. Tulpan, H. Hoos, A. Condon, "Stochastic local search algorithms for DNA word design", Proc. 8th DIMACS Workshop on DNA Based Computers, June 2002.

Time-Varying Distributed H Systems with Parallel Computations: The Problem Is Solved*

Maurice Margenstern[1], Yurii Rogozhin[2], and Sergey Verlan[1]

[1] LITA, Université de Metz, France
{margens,verlan}@sciences.univ-metz.fr
[2] IMI, Academy of Sciences of Moldova
rogozhin@math.md

Abstract. In this article we show that time-varying distributed H systems (TVDH systems) with one component are able to model any type-0 grammar. Thus we completely answered to the question of constructing TVDH systems of smallest degree which generate any RE language using the parallel nature of molecular computations based on splicing operations. Another interesting point is that the proof is based on a simulation of a TVDH system of degree two and not of type-0 grammars as it is usually done in similar proofs.

1 Introduction

Starting from [6], a grounding paper on splicing computations, a lot of studies were devoted to various extensions of H systems originating from [1], in particular, pointing at their possible universality power. One of these models are time-varying distributed H systems which were recently introduced in [7, 8] by G. Păun. This model introduces *components,* later see the formal definition, which cannot all be used at the same time but one after another, periodically.

In [7], it is proved that 7 different *components* are enough in order to generate any recursively enumerable language. In [3], the first two present authors proved that one component is enough in order to generate any recursively enumerable language. The proof was made by a sequential modelling of Turing machines.

In the meanwhile, A. Păun published a paper [5] where he proved that time-varying distributed H systems with 4 components generate all recursively enumerable languages. But his solution is a parallel one: its proof consists in modelling any type-0 grammar. A. Păun's result was improved in 2000 by the first two authors of the present paper by reducing the number of components of TVDH systems which model type-0 formal grammars down to 3 [2]. In 2002 the authors proved that it is possible to model type-0 grammars only with two components [4].

* Work supported by French Ministry of Education, *NATO* project PST.CLG.976912 and project IST-2001-32008 **MolCoNet**

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 48–54, 2004.

Now we improve the last result by reducing the number of components of TVDH systems which model type-0 formal grammars down to one. Thus we completely solve the problem of constructing TVDH systems of smallest degree which generate any RE language using the parallel nature of molecular computations based on splicing operations.

At last, but not the least, we notice that this proof is very different from previous ones of the first two authors. Indeed, in the mentioned previous proofs, the authors simulated a type-0 grammar. In this proof, we simulate a TVDH system of degree 2, D_G^2, from [4]. Moreover, the simulation is very close to the execution of D_G^2: most applications of a rule of D_G^2 is simulated in one step of the TVDH system of degree 1 which is constructed in the paper. In this regard, this proof is more direct than the previous ones.

Within the limits of this paper it is not possible to give a thorough checking of the computations. We only outline the main arguments of the proof. In [9] the reader can find a similar method in action which is thoroughly explained.

2 Basic Definitions

We recall some notions. An *(abstract) molecule* is simply a word over some alphabet. A *splicing rule* (over alphabet V), is a quadruple (u_1, u_2, u_1', u_2') of words $u_1, u_2, u_1', u_2' \in V^*$, which is often written in a two dimensional way as follows: $\frac{u_1 | u_2}{u_1' | u_2'}$.

A splicing rule $r = (u_1, u_2, u_1', u_2')$ is applicable to two molecules m_1, m_2 if there are words $w_1, w_2, w_1', w_2' \in V^*$ with $m_1 = w_1 u_1 u_2 w_2$ and $m_2 = w_1' u_1' u_2' w_2'$, and produces two new molecules $m_1' = w_1 u_1 u_2' w_2'$ and $m_2' = w_1' u_1' u_2 w_2$. In this case, we also write $(m_1, m_2) \vdash_r (m_1', m_2')$.

A pair h = *(V,R),* where V is an alphabet and R is a finite set of splicing rules, is called an *splicing scheme* or an H *scheme.*

For an H scheme h = *(V, R)* and a language $L \subseteq V^*$ we define:

$\sigma_h(L) = \sigma_{(V,R)}(L) \stackrel{\text{def}}{=} \{w, w' \in V^* | \exists w_1, w_2 \in L : \exists r \in R : (w_1, w_2) \vdash_r (w, w')\}$.

A time-varying distributed H *system* [7] (of degree n, $n \geq 1$), (TVDH system) is a construct:

$$D = (V, T, A, R_1, R_2, \ldots, R_n),$$

where V is *an alphabet,* $T \subseteq V$, the *terminal alphabet,* $A \subseteq V^*$, a finite set of *axioms,* and *components* R_i are finite sets of splicing rules over V, $1 \leq i \leq n$.

At each moment $k = n \cdot j + i$, for $j \geq 0$, $1 \leq i \leq n$, only component R_i is used for splicing the currently available strings. Specifically, we define

$$L_1 = A; \qquad L_{k+1} = \sigma_{h_i}(L_k), \text{ for } i \equiv k(mod\ n),\ h_i = (V, R_i).$$

Therefore, from a step k to the next step, $k+1$, one passes only the result of splicing the strings in L_k according to the rules in R_i for $i \equiv k(mod\ n)$; the strings in L_k which cannot enter a splicing are removed.

The language generated by D is:

$$L(D) \stackrel{\text{def}}{=} (\cup_{k\geq 1} L_k) \cap T^*.$$

3 TVDH Systems of Degree 1

Theorem 1. *For any type-0 formal grammar $G = (N, T, P, S)$ there is a TVDH system $D_G = (V, T, A, R_1)$ of degree 1 which simulates G and $L(D_G) = L(G)$.*

The theorem follows immediately from Lemmas 1 and 2.

Lemma 1. *For any type-0 formal grammar $G = (N, T, P, S)$ there is a TVDH system $D_G^2 = (V, T, A, R_1, R_2)$ of degree 2 which simulates G and $L(D_G) = L(G)$.*

The proof of this lemma is given in [4].

Lemma 2. *For any TVDH system D_G^2 constructed in Lemma 1 there is a TVDH system $D_G = (V, T, A, R_1)$ of degree 1 which simulates D_G^2 and $L(D_G) = L(D_G^2)$.*

In order to prove the lemma we slightly transform, from technical considerations, the TVDH2 system D_G^2 from [4] into a form which is more suitable for a simulation. Some letters are numbered and some rules have a bigger context, but these transformations do not alter neither the functioning of the system nor the generated language. We will not distinguish between this transformed system and the system from [4] and we will refer further to it as D_G^2. In fact, this system contains the rules 1–18 (see below) distributed over two components as well as some additional rules. The system has the following properties:

- It has two components
- The system has two independent sets of axioms and they persist during the computations. One of these sets is used for splicing in the first component only, the other one is used for splicing in the second component only
- The rules are made in a such way that molecules that code different stages of the simulation of the grammar can be spliced only with axioms from the two above sets.

The method we use to perform the simulation of the system D_G^2 is the following. In our system D_G we have two versions of each axiom for each set: a normal (active) and a prime (passive) version. On an odd step of computation we have active versions of molecules from the first set and passive versions of molecules from the second set. On an even step we have passive versions of the molecules from the first set and active version of the molecules from the second one. The prime versions of molecules cannot enter any rule except the unpriming rule. The normal versions of molecules may enter either a splicing rule which corresponds to a rule of D_G^2 or they may enter a priming rule which tags these molecules with a prime. Thus we create the active molecules only at the moment we really need them. In that way we simulate the behaviour of the first component during an

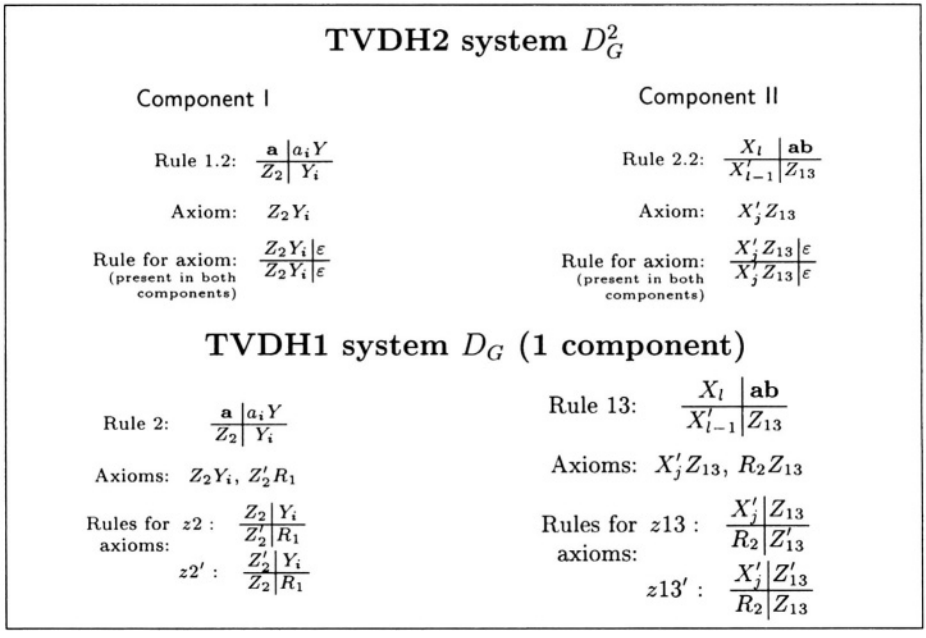


Fig. 1. Transformation of TVDH2 system D_G^2 into a TVDH1 system.

odd step and we simulate the behaviour of the second component during an even step. And so, we perform a correct simulation. The rules from both components of D_G^2 are simply merged into one component. We start with a normal version of the first set and with a prime version of the second set. The transformation of TVDH2 system into TVDH1 system is shown in Fig. 1.

For example: we have molecule Z_2Y_i on a odd step. It may be used for splicing using rule 2 thus we performed the simulation of rule 1.2 of the TVDH2 system or it may be tagged with a prime using rule $z2$: $(Z_2Y_i, Z'_2R_1) \vdash_{z2} (Z'_2Y_i, Z_2R_1)$. On the next step (even step) this molecule (Z_2Y_i) is not accessible so rule 2 cannot be applied. On the other hand we can unprime molecule Z'_2Y_i using rule $z2'$ (molecule Z_2R_1 was obtained on the previous step) and thus we obtain again molecule Z_2Y_i and we are on a odd step.

A similar lot happens to all molecules having Z (with indices).

Rules 1.8, 1.10, 2.7 and 2.9 are used in D_G^2 to obtain the resulting terminal string. In D_G, due to technical details, they are simulated in a special way by rules 7, 16, x.1, x.2, and r.1–r.16.

Now we will give a more formal definition of the system.

We define $D_G = (V, T, A, R_1)$ as follows.

Let $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$.

In what follows we will assume the following:

$1 \leq i \leq n,\ 1 \leq j \leq n-1,\ 2 \leq k \leq n,\ s \in \{0, 2\},\ \mathbf{a}, \mathbf{b} \in N \cup T \cup \{B\}$.

Alphabet: $V = \{V^2 \setminus \{C_1, C_2, D_1, D_2\}\} \cup \{R_1, R_2, Z'_{m\ (1\le m\le 18)}\} \cup \{C_1^s, C_2^s, R, Z_7^p, Z_{16}^p, Z_D^p, Z_{E\ (1\le p\le 3)}^p, D_1^s, E_2^s, Z_D, Z_E\}$.

The terminal alphabet is $T = T^2$ (the same as for G).

Axioms: $A = \{XSBY\} \cup A_1 \cup A_2 \cup A_R$, where

$A_1 = \{Z_1 v Y_i : \exists u \to v a_i \in P\} \cup \{Z_1 Y_i : \exists u \to \epsilon \in P\} \cup$
$\{Z_2 Y_i, Z_3 Y_j, Z_4 Y'_j, Z_5 Y''_j, Z_6 Y, X' Z_8, X'' Z_9\} \cup \{Z'_m R_1 (1 \le m \le 7), R_1 Z'_8, R_1 Z'_9\}$.

$A_2 = \{X_i a_i Z'_{11}, X_j Z'_{12}, X'_j Z'_{13}, X''_j Z'_{14}, X Z'_{15}, Z'_{17} Y', Z'_{18} Y''\} \cup$
$\{Z_{17} R_2, Z_{18} R_2, R_2 Z_{m(11\le m\le 16)}\}$.

$A_R = \{C_1^0 Z_{16}, C_1^2 Z_{16}^2, R Z_{16}^1, R Z_{16}^3, Z_7^1 C_2^0\} \cup$
$\{Z_7^3 C_2^2, Z_7^2 R, Z_7 R, D_1^0 Z_D^1, D_1^2 Z_D^3, R Z_D^2, R Z_D, Z_E E_2^0, Z_E^2 E_2^2, Z_E^1 R, Z_E^3 R\}$.

The axioms from A_1, A_2 are used to simulate the behaviour of the first and respectively second component. The axioms from A_R are used in order to obtain the result.

Simulation of rules of the first component of the original system (the original number is given in parenthesis):

$$1(1.1) : \frac{\mathbf{a}\,|\,uY}{Z_1\,|\,vY_i}\ (\exists u\to va_i\in P); \quad 1'(1.1') : \frac{\mathbf{a}\,|\,a_iuY}{Z_1\,|\,Y_i}\ (\exists u\to\epsilon\in P); \quad 2(1.2) : \frac{\mathbf{a}\,|\,a_iY}{Z_2\,|\,Y_i};$$

$$3(1.9) : \frac{\mathbf{a}\,|\,Y''_j}{Z_3\,|\,Y_j}; \quad 4(1.3) : \frac{\mathbf{ab}\,|\,Y_k}{Z_4\,|\,Y'_{k-1}}; \quad 5(1.4) : \frac{\mathbf{a}\,|\,Y'_j}{Z_5\,|\,Y''_j};$$

$$6(1.7) : \frac{\mathbf{a}\,|\,Y''}{Z_6\,|\,Y}; \quad 7(1.8) : \frac{\mathbf{a}\,|\,BY''}{Z_7\,|\,C_2^s}; \quad 8(1.5) : \frac{X_1\,|\,\mathbf{ab}}{X'\,|\,Z_8}; \quad 9(1.6) : \frac{X'\,|\,\mathbf{a}}{X''\,|\,Z_9};$$

Simulation of rules of the second component of the original system (the original number is given in parenthesis):

$$11(2.1) : \frac{X\,|\,\mathbf{a}}{X_i a_i\,|\,Z_{11}}; \quad 12(2.8) : \frac{X''_j\,|\,\mathbf{a}}{X_j\,|\,Z_{12}}; \quad 13(2.2) : \frac{X_k\,|\,\mathbf{ab}}{X'_{k-1}\,|\,Z_{13}};$$

$$14(2.3) : \frac{X'_j\,|\,\mathbf{a}}{X''_j\,|\,Z_{14}}; \quad 15(2.6) : \frac{X''\,|\,\mathbf{a}}{X\,|\,Z_{15}}; \quad 16(2.7) : \frac{X''\,|\,\mathbf{a}}{C_1^s\,|\,Z_{16}};$$

$$17(2.4) : \frac{\mathbf{ab}\,|\,Y_1}{Z_{17}\,|\,Y'}; \quad 18(2.5) : \frac{\mathbf{a}\,|\,Y'}{Z_{18}\,|\,Y''};$$

Creation of axioms from the first set (used in the first component):
Priming rules:

$$z1 : \frac{Z_1\,|\,vY_i}{Z'_1\,|\,R_1}; \quad z2 : \frac{Z_2\,|\,Y_i}{Z'_2\,|\,R_1}; \quad z3 : \frac{Z_3\,|\,Y_j}{Z'_3\,|\,R_1}; \quad z4 : \frac{Z_4\,|\,Y'_j}{Z'_4\,|\,R_1};$$

$$z5 : \frac{Z_5\,|\,Y''_j}{Z'_5\,|\,R_1}; \quad z6 : \frac{Z_6\,|\,Y}{Z'_6\,|\,R_1}; \quad z8 : \frac{X'\,|\,Z_8}{R_1\,|\,Z'_8}; \quad z9 : \frac{X''\,|\,Z_9}{R_1\,|\,Z'_9};$$

Unpriming rules:

$$z1' : \frac{Z'_1\,|\,vY_i}{Z_1\,|\,R_1}; \quad z2' : \frac{Z'_2\,|\,Y_i}{Z_2\,|\,R_1}; \quad z3' : \frac{Z'_3\,|\,Y_j}{Z_3\,|\,R_1}; \quad z4' : \frac{Z'_4\,|\,Y'_j}{Z_4\,|\,R_1};$$

$$z5' : \frac{Z'_5\,|\,Y''_j}{Z_5\,|\,R_1}; \quad z6' : \frac{Z'_6\,|\,Y}{Z_6\,|\,R_1}; \quad z8' : \frac{X'\,|\,Z'_8}{R_1\,|\,Z_8}; \quad z9' : \frac{X''\,|\,Z'_9}{R_1\,|\,Z_9};$$

Creation of axioms from the second set (used in the second component):
Unprimingrules:

$$z11' : \frac{X_i a_i \mid Z'_{11}}{R_2 \mid Z_{11}} ; \quad z12' : \frac{X_j \mid Z'_{12}}{R_2 \mid Z_{12}} ; \quad z13' : \frac{X'_j \mid Z'_{13}}{R_2 \mid Z_{13}} ; \quad z14' : \frac{X''_j \mid Z'_{14}}{R_2 \mid Z_{14}} ;$$

$$z15' : \frac{X \mid Z'_{15}}{R_2 \mid Z_{15}} ; \quad z17' : \frac{Z'_{17} \mid Y'}{Z_{17} \mid R_2} ; \quad z18' : \frac{Z'_{18} \mid Y''}{Z_{18} \mid R_2} ;$$

Priming rules:

$$z11 : \frac{X_i a_i \mid Z_{11}}{R_2 \mid Z'_{11}} ; \quad z12 : \frac{X_j \mid Z_{12}}{R_2 \mid Z'_{12}} ; \quad z13 : \frac{X'_j \mid Z_{13}}{R_2 \mid Z'_{13}} ; \quad z14 : \frac{X''_j \mid Z_{14}}{R_2 \mid Z'_{14}} ;$$

$$z15 : \frac{X \mid Z_{15}}{R_2 \mid Z'_{15}} ; \quad z17 : \frac{Z_{17} \mid Y'}{Z'_{17} \mid R_2} ; \quad z18 : \frac{Z_{18} \mid Y''}{Z'_{18} \mid R_2} ;$$

Rules for result:

$$x.1 : \frac{\mathbf{a} \mid C_2^s}{Z_E E_2^s \mid \varepsilon} ; \quad x.2 : \frac{C_1^s \mid \mathbf{a}}{\varepsilon \mid D_1^s Z_D} ;$$

$$r.1 : \frac{C_1^s \mid Z_{16}}{R \mid Z_{16}^1} ; \quad r.2 : \frac{C_1^s \mid Z_{16}^1}{R \mid Z_{16}^2} ; \quad r.3 : \frac{C_1^s \mid Z_{16}^2}{R \mid Z_{16}^3} ; \quad r.4 : \frac{C_1^s \mid Z_{16}^3}{R \mid Z_{16}} ;$$

$$r.5 : \frac{D_1^s \mid Z_D}{R \mid Z_D^1} ; \quad r.6 : \frac{D_1^s \mid Z_D^1}{R \mid Z_D^2} ; \quad r.7 : \frac{D_1^s \mid Z_D^2}{R \mid Z_D^3} ; \quad r.8 : \frac{D_1^s \mid Z_D^3}{R \mid Z_D} ;$$

$$r.9 : \frac{Z_7 \mid C_2^s}{Z_7^1 \mid R} ; \quad r.10 : \frac{Z_7^1 \mid C_2^s}{Z_7^2 \mid R} ; \quad r.11 : \frac{Z_7^2 \mid C_2^s}{Z_7^3 \mid R} ; \quad r.12 : \frac{Z_7^3 \mid C_2^s}{Z_7 \mid R} ;$$

$$r.13 : \frac{Z_E \mid E_2^s}{Z_E^1 \mid R} ; \quad r.14 : \frac{Z_E^1 \mid E_2^s}{Z_E^2 \mid R} ; \quad r.15 : \frac{Z_E^2 \mid E_2^s}{Z_E^3 \mid R} ; \quad r.16 : \frac{Z_E^3 \mid E_2^s}{Z_E \mid R} ;$$

Program Check

The obtained system was checked for errors using `TVDHsim`: time varying distributed H systems simulator, a program developed by the third author at the University of Metz. It can be found at `http://lita.sciences.univ-metz.fr/~verlan/`. The same program was also widely used during the construction of the system.

References

[1] Head, T.: Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors. *Bulletin of Mathematical Biology* **49** (1987) 737–759.

[2] Margenstern M., Rogozhin Yu.: About Time-Varying Distributed H Systems. *Lecture Notes in Computer Science,* Springer, vol. **2054,** 2001, p.53-62.

[3] Margenstern, M., Rogozhin, Yu.: Time-varying distributed H systems of degree 1 generate all recursively enumerable languages, in Words, Semigroups and Transductions (M. Ito, Gh. Paun, S. Yu, eds), World Sci.,Singapore, 2001, p. 329-340.

[4] Margenstern, M., Rogozhin, Yu., Verlan, S.: Time-Varying Distributed H Systems of Degree 2 Can Carry Out Parallel Computations. *Lecture Notes in Computer Science,* Springer, vol. **2568,** 2003, p.326-336.
[5] Păun, A.: On Time-Varying H Systems. *Bulletin of EATCS.* **67** (1999) 157–164.
[6] Păun, G., Rozenberg, G., Salomaa, A.: Computing by splicing. *TCS.* **168,** no.2 (1996) 321–336.
[7] Păun, G.: DNA Computing Based on Splicing: Universality Results. *TCS.* **231,** no.2 (2000) 275–296.
[8] Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing: New Computing Paradigms. Springer (1998).
[9] Verlan, S.: A frontier result on enhanced time-varying distributed H systems with parallel computations, Preproceedings of DCFS'03, Descriptional Complexity of Formal Systems, Budapest, Hungary, July 12-14, 2003, p. 221-232.

Deadlock Decidability in Partial Parallel P Systems*

Daniela Besozzi[1], Giancarlo Mauri[1,2], and Claudio Zandron[1,2]

[1] Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39, 20135 Milano, Italy
besozzi@dico.unimi.it
[2] Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8, 20136 Milano, Italy
{mauri, zandron}@disco.unimib.it

Abstract. In parallel rewriting P systems, the notion of deadlock is used to describe situations where evolution rules with different target indications are simultaneously applied on a common string. In this paper we claim that the generative power of *partial parallel P systems* (*PPP,* in short) with deadlock is equivalent to matrix grammars without appearance checking, and we prove that it is decidable whether or not a *PPP* will ever reach a deadlock configuration.

1 Introduction

We assume that the reader is familiar with P systems [7] (see also the web address http://psystems.disco.unimib.it/). In this paper we consider *rewriting P systems* [8] and our aim is to extend the application of evolution rules from sequential rewriting to the parallel one (as in [6]). This fact is also biological motivated, as a cellular substance could be processed by many chemical reactions (each on a different site) at the same time.

The use of parallel rewriting means that a string has to be simultaneously processed, if possible, by more than one rule, according to the prescribed parallel rewriting method. So, in parallel rewriting P systems there are three levels of parallelism, involving membranes, objects and rules. On the other hand, if the rules we apply on the same string have different target indications, then we have consistency problems for the communication of the resulting string, as there are contradictory indications about the region where the string should be at the next step. This problem has been previously faced and solved with different strategies [6, 3], here we follow the approach introduced in [2]: we say that when rules with mixed target indications are applied at the same time on a common string, then a *deadlock state* occurs inside the system. When a situation of deadlock

* Work partially supported by contribution of EU commission under The Fifth Framework Programme, project "MolCoNet" IST-2001-32008.

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 55–60, 2004.

arises for a string, then it is not sent to outer or inner regions but it remains inside the current membrane, though it will not be processed anymore by any other rule. Hence the deadlock state for that string causes its further processing and communication to be stopped.

In [2, 4] we began the analysis of parallel rewriting P systems with and without deadlock, which use different types of parallelism. We continue here the analysis of P systems with deadlock which use partial parallel rewriting, characterizing their computational power and also proving that it is decidable whether a system of this type will ever reach a deadlock configuration.

2 Partial Parallel P Systems with Deadlock

We refer to [8] for a formal definition of rewriting P systems. Here we explain how the application of evolution rules is extended to *partial parallel rewriting:* as always, in one step all regions are processed simultaneously by using the rules in a nondeterministic and maximally parallel manner. Moreover, at each step of a computation a string has to be simultaneously processed, if possible, by no more than k rules (for a fixed $k \geq 2$). That is, given an alphabet V, a string $w = x_1a_1x_2a_2x_3a_3 \dots x_ka_kx_{k+1}$ (with $x_i \in V^*$ and $a_i \in V, i = 1, \dots, k$) and some rules $r_1 : a_1 \to \alpha_1, \dots, r_k : a_k \to \alpha_k$ (not necessarily distinct), then in one step the new string $w' = x_1\alpha_1x_2\alpha_2x_3\alpha_3 \dots x_k\alpha_kx_{k+1}$ is obtained. Hence, in one step *exactly* k symbols are substituted in the string w, but if the string contains less than k symbols which can be the subject of a rewriting rule, then the parallel rewriting step is blocked. Of course, if more than k symbols can be the subject of some evolution rules, then only k will be nondeterministically chosen out of them and simultaneously rewritten by means of $\bar{k}$ rules, where it holds $\bar{k} = k$ if the applied rules are all different, $\bar{k} < k$ if some rule is used to rewrite many occurrences of the same symbol.

When we simultaneously apply two or more rules to the same string, we have to check that their target indications match before communicating the resulting string to the right region. To this aim, for every region $i = 0, \dots, n$ of the membrane structure we divide the set R_i of evolution rules into mutually disjoint subsets of rules which have the same target indications, that is $R_i = \mathcal{H}_i \cup \mathcal{O}_i \cup \mathcal{I}_i$, where $\mathcal{H}_i = \{r \in R_i \mid tar(r) = here\}, \mathcal{O}_i = \{r \in R_i \mid tar(r) = out\}$ and $\mathcal{I}_i = \{r \in R_i \mid tar(r) = in\}$.

Consider now a set of rules $\mathcal{R} = \{r_1, \dots, r_k\}$ all of which could be applied on a string w. If it holds that (1) $\mathcal{R} \subseteq \mathcal{H}_i$ or (2) $\mathcal{R} \subseteq \mathcal{I}_i$ or (3) $\mathcal{R} \subseteq \mathcal{O}_i$, that is all targets match, then the resulting string w' (1) remains inside the current region i, (2) is communicated to a (nondeterministically chosen) inner region, (3) is communicated to the outer region. Otherwise, if $\mathcal{R}$ contains rules with mixed target indications (that is, e.g., $\mathcal{R} \cap \mathcal{H}_i \neq \emptyset \wedge \mathcal{R} \cap \mathcal{I}_i \neq \emptyset$), then we have consistency problems for the communication of the resulting string, as there are contradictory indications about the region where the string should be at the next step. When rules with different target indications are applied at the same time on a common string, we say that we have a **deadlock state** inside the system.

The string is not sent to outer or inner regions but it remains inside the current membrane, though its further processing and communication are stopped. When a deadlock state occurs inside a membrane, for the other strings we can assume that: (D_1) they can enter that membrane, be processed by local rules and even exit the region (if they are not in a deadlock state after the application of local rules). Alternatively, we can assume that the deadlock state propagates to the entire membrane where such string is placed, i.e.: (D_2) other strings can enter the membrane and be processed by local rules, but they can never exit the region even if they are not in a deadlock state.

In the first case, (D_1), the membrane acts like a *filter* for wrong or right strings, that is for strings with or without deadlock, stopping the wrong ones and letting the right ones proceed. A wrong string could be seen as an error taking place during the computation of the P system, and hence it must be stopped. This interpretation is considered in the following. In the second case, (D_2), it happens that a consistency problem for target matching on a single string causes the system to lose an entire computing unit, as no strings are allowed to exit that membrane anymore. This interpretation differs, for example, from the dissolving action δ of membranes, in fact in this case the membrane is lost but its objects are recovered in the outer region, while for deadlock membrane both the membrane and its objects are lost.

A generic configuration $C = (\mu, M_0, \ldots, M_n)$ is said to be *free* if there are no deadlock states inside the system at that time. Otherwise, we say that the system is in a *deadlock configuration,* and we denote by $\langle M_j \rangle$ all languages which contain at least a deadlocked string. A transition starting from a deadlock configuration will always reach another deadlock configuration, that is we do not consider the possibility of removing deadlock states.

A configuration where all membranes are in a deadlock state is said to be a *global deadlock* configuration, otherwise we talk about *local deadlock* configurations. A sequence of transitions of free and (local) deadlock configurations forms a *computation.* Observe that if the P system is processing a single string and if there are no rules which can increase the number of the strings (both conditions are considered in the following), then a local deadlock configuration causes the computation to halt.

We will denote by $EParRP((P)_{=k}, D)$ the family of languages generated by extended *PPP,* where $(P)_{=k}$ denotes the partial parallel method (with the rewriting of exactly k symbols) and D denotes the possibility of having deadlock states.

3 Deadlock Decidability

We refer to [5] for a formal definition and results about matrix grammars without appearance checking. Here we only recall that: (i) MAT is closed under the operation of intersection with regular languages: given any $L \in MAT$ and any $L' \in REG$, it holds that $L \cap L' \in MAT$; *(ii)* the emptiness problem for languages

in *MAT* is decidable: given any matrix grammar G, it is decidable whether or not $L(G) = \emptyset$.
With respect to *PPP*, it is possible to show [1] that:

Theorem 1. $EParRP((P)_{=k}, D) = MAT$.

Given the results above and the computational equivalence among *MAT* and *PPP* with deadlock, we can now prove that:

Theorem 2. *It is decidable whether or not a partial parallel rewriting P system (where exactly k rules are to be applied at each computing step) will ever reach a deadlock configuration.*

Proof. Consider a system $\Pi = (V, T, \mu, M_0, \ldots, M_n, R_0, \ldots, R_n)$ such that $L(\Pi) \in EParPRP((P)_{=k}, D)$. First, we show how to construct a matrix grammar $G = (N, T, S, P)$, which generates the same language as Π. The alphabet is $N = V \cup \{S, \#, \dagger\} \cup \{\bar{A} \mid A \in V\} \cup \{X_i \mid i = 0, \ldots, n\}$, where $V, \{S, \#, \dagger\}, \{\bar{A} \mid A \in V\}, \{X_i \mid i = 0, \ldots, n\}$ are mutually disjoint sets.

For each membrane m_i in μ, for any $i \in \{0, \ldots, n\}$, we define a corresponding set of matrices of the following four types:
1. Starting matrices: for each string $w \in M_i$, we define a matrix of the form $[S \rightarrow X_i w]$.
2. Simulation matrices: let $lab_{i(out)}, lab_{i(in)}$ be the sets of labels of the membranes placed outside and inside (if any) the membrane m_i. To simulate the application of rules which causes a deadlock state, we define the matrices of the form $[X_i \rightarrow \dagger, A_1 \rightarrow \bar{\alpha}_1, \ldots, A_k \rightarrow \bar{\alpha}_k]$, for $A_1 \rightarrow \alpha_1(tar), \ldots, A_k \rightarrow \alpha_k(tar)$ rules in R_i with mixed target indications $(tar \in \{here, in, out\})$.
Instead, the matrices that simulate the application of rules with equal targets are: $[X_i \rightarrow X_j, A_1 \rightarrow \bar{\alpha}_1, \ldots, A_k \rightarrow \bar{\alpha}_k]$, where $A_1 \rightarrow \alpha_1(tar), \ldots, A_k \rightarrow \alpha_k(tar)$ are rules in $\mathcal{H}_i$ or $\mathcal{O}_i$ or $\mathcal{I}_i$, and it holds $j = i$ if $tar = here$, $j \in lab_{i(out)}$ if $tar = out$ (with the condition that, if $i = 0$, then $X_j = \lambda$), and $j \in lab_{i(in)}$ if $tar = in$.
3. Checking matrices: $[X_i \rightarrow X_i, \bar{A} \rightarrow A]$ for all $\bar{A} \in N$.
4. Trap matrices: $[\bar{A} \rightarrow \#]$ for all $\bar{A} \in N$.

It is easy to show that $L(G) = L(\Pi)$: at the first step of a derivation, only a starting matrix can be used for rewriting the axiom S. The choice of the starting matrix determines both the string and the membrane whose evolution and rules will be simulated. At the second step of a derivation, only a simulation matrix can be used. In fact, we recall that it is not possible to skip any rule in a matrix, hence in this case nor a checking matrix nor a trap matrix could be used, as they both contain rules for overlined symbols (which do not appear in the string yet). Moreover, if no rules can be applied inside a membrane, then the corresponding matrices cannot be used in the grammar, so the derivation is blocked. Observe that there are as many simulation matrices as all possible $\binom{m+k-1}{k}$ combinations of k rules (with equal or different target indications) from any set R_i are (with $|R_i| = m$); in this way we can simulate the nondeterministic application of exactly k rules inside any membrane m_i of Π.

Assume a simulation matrix is going to be used. According to the targets of the rules appearing in the matrix, the symbol X_i can be rewritten as: (1) the symbol †, if the targets are mixed; (2) the symbol X_i, if all targets are equal to *here;* (3) the symbol X_j, for $j \in lab_{i(out)} \cup lab_{i(in)}$, if all targets are equal to *out* or *in.*

In the case (1), the symbol † will never be erased and the string will not belong to *L(G),* so we can correctly simulate a deadlock state in Π. In the cases (2) and (3), the new support symbol determines which matrices can be applied at the following steps, that is only the matrices which correctly simulate the evolution of the corresponding string in Π. In all cases, the k rules $A_1 \rightarrow \bar{\alpha}_1, \ldots, A_k \rightarrow \bar{\alpha}_k$ simulate the parallel rewriting of the current string. The strings $\bar{\alpha}_l$, for $l = 1, \ldots, k$, correspond to the strings α_l, where all symbols from *V* have been substituted with the corresponding overlined symbols in *N.* We point out that, in a matrix, the rules are not applied in parallel but one after the other, so we have to use overlined symbols in order to avoid a wrong simulation of rules in Π.

After a simulation matrix has been used, only a checking matrix or a trap matrix can be used. If we use a trap matrix, then the trap symbol # is introduced and never removed. The only way to continue a derivation and to obtain a terminal string is to use some checking matrices in sequence. Only if at least k overlined symbols are substituted with the corresponding non-overlined symbols, then at the next step we can use another simulation matrix (such a process roughly corresponds to a nondeterministic choice of rules in Π). The remaining overlined symbols (if any) can be substituted at the following derivations steps. If this does not happen, in any moment a trap matrix can be used and the symbol # is introduced. Assume now that we have simulated the application of rules in $\mathcal{O}_0$, the symbol X_0 is erased and we have to check whether or not the current string is terminal. If it contains some overlined symbols, then the trap symbol is introduced by means of one or more trap matrices. Otherwise, no other matrices can be used and the string will belong to *L(G).* It follows that $L(G) = L(\Pi)$.

Now, to prove the theorem, we need to add to G the following **deadlock matrices:** $[\dagger \rightarrow \dagger, \bar{A} \rightarrow \lambda]$, for all $A \in V \setminus T$, and $[\dagger \rightarrow \dagger, \bar{a} \rightarrow a]$, for all $a \in T$.

We obtain a new grammar G' which generates the language *L(G)* plus some new strings $w \in \{\dagger\}T^*$. In fact, after a simulation matrix (corresponding to a deadlock state in Π) has been used in *G,* by means of the deadlock matrices of G' we generate some new strings where all nonterminal overlined symbols are deleted, while the terminal overlined symbols are substituted with the corresponding non-overlined terminal symbols. Observe that the deadlock matrices do not modify any other original derivation in *G*, because they can be applied only over the strings which contain the symbol †.

Consider now the language $\mathcal{L} = L(G') \cap L_\dagger$, where $L_\dagger = \{w = \dagger x \mid x \in T^*\}$. As $L_\dagger \in REG$ and the family *MAT* is closed under intersection with regular languages, it follows that also $\mathcal{L}$ belongs to *MAT.* The language $\mathcal{L}$ is non–empty if and only if the symbol † appears in at least one string of $L(G')$, that is if and only if at least one deadlock state occurs in the system. As the emptiness problem

is decidable for the family of languages *MAT,* it holds that it is also decidable to state whether or not a *PPP* will ever reach a deadlock configuration. □

Corollary 1. *Given a partial parallel P system (where exactly k rules are to be applied at each computing step), it is decidable to determine in which membrane a deadlock state will ever occur.*

Proof. Consider the matrix grammar G' constructed in Theorem 2, and let us modify it in order to determine in which membrane of the system a deadlock state occurs. It suffices to use rules of the form $X_i \rightarrow \dagger_i$ in the simulation matrices, and rules $\dagger_i \rightarrow \dagger_i$ in the deadlock matrices. The subscript $i, i = 0, \ldots, n-1$, obviously corresponds to the label of the membrane whose evolution rules are simulated. The language $\mathcal{L} = L(G') \cap (\bigcup_{i=0}^{n-1} \{\dagger_i\} T^*)$ is still in *MAT,* and the previous conclusions hold. □

4 Final Remarks

The computational equivalence between *PPP* with deadlock and matrix grammars without appearance checking enables to prove that, for P systems where an exact fixed number of rules are simultaneously applied, the reaching of deadlock states is a decidable problem. It is still an open problem to establish whether the deadlock decidability is solvable when other parallel rewriting methods are used (see, e.g., [4]).

Another important research topic is concerned with finding out any possible biological counterpart of deadlock, in order to understand if P systems with deadlock are suitable to model real situations, for example in case of cellular malfunctions or disease.

References

[1] D. Besozzi, *Computational and Modelling Power of P Systems,* PhD Thesis, 2003.
[2] D. Besozzi, C. Ferretti, G. Mauri, C. Zandron, Parallel Rewriting P Systems with Deadlock, *Proceedings of 8th International Workshop on DNA Based Computers (M. Hagiya, A. Ohuchi, eds.),* Springer, LNCS 2568, 302–314, 2003.
[3] D. Besozzi, G. Mauri, C. Zandron, Parallel Rewriting P Systems without Target Conflicts, *Proceedings of Membrane Computing International Workshop WMC-CdeA2002 (G. Paun, G. Rozenberg, A. Salomaa, C. Zandron, eds.),* Springer, LNCS 2597, 119–133, 2003.
[4] D. Besozzi, C. Ferretti, G. Mauri, C. Zandron, P Systems with Deadlock, *Biosystems,* 70, 2, July 2003, 95–105 .
[5] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory,* Springer-Verlag, Berlin, 1989.
[6] S.N. Krishna, *Languages of P systems: Computability and Complexity,* PhD Thesis, 2001.
[7] G. Păun, *Membrane Computing. An Introduction,* Springer-Verlag, Berlin, 2002.
[8] C. Zandron, *A Model for Molecular Computing: Membrane Systems,* PhD Thesis, 2001.

Languages of DNA Based Code Words

Nataša Jonoska and Kalpana Mahalingam

University of South Florida, Department of Mathematics,
Tampa, FL 33620, USA
jonoska@math.usf.edu
mahaling@helios.acomp.usf.edu

Abstract. The set of all sequences that are generated by a biomolecular protocol forms a language over the four letter alphabet $\Delta = \{A, G, C, T\}$. This alphabet is associated with a natural involution mapping θ, $A \mapsto T$ and $G \mapsto C$ which is an antimorphism of Δ^*. In order to avoid undesirable Watson-Crick bonds between the words (undesirable hybridization), the language has to satisfy certain coding properties. In this paper we build upon the study initiated in [11] and give necessary and sufficient conditions for a finite set of "good" code words to generate (through concatenation) an infinite set of "good" code words with the same properties. General methods for obtaining sets of "good" code words are described. Also we define properties of a splicing system such that the language generated by the system preserves the desired properties of code words.

1 Introduction

In bio-molecular computing and in particular DNA based computations and DNA nanotechnology, one of the main problems is associated with the design of the oligonucleotides such that mismatched pairing due to the Watson-Crick complementarity is minimized. In laboratory experiments non-specific hybridizations pose potential problems for the results of the experiment. Many authors have addressed this problem and proposed various solutions. Common approach has been to use the Hamming distance as a measure for uniqueness [1, 5, 6, 8, 15]. Deaton et al. [5, 8] used genetic algorithms to generate a set of DNA sequences that satisfy predetermined Hamming distance. Marathe et al. [16] also used Hamming distance to analyze combinatorial properties of DNA sequences, and they used dynamic programing for design of the strands used in [15]. Seeman's program [19] generates sequences by testing overlapping subsequences to enforce uniqueness. This program is designed for producing sequences that are suitable for complex three-dimensional DNA structures, and the generation of suitable sequences is not as automatic as the other programs have proposed. Feldkamp et al. [7] also uses the test for uniqueness of subsequences and relies on tree structures in generating new sequences. Ruben at al. [18] use a random generator for initial sequence design, and afterwards check for unique subsequences with a predetermined properties based on Hamming distance. One of the first theoretical observations about number of DNA code words satisfying minimal

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 61–73, 2004.

Hamming distance properties was done by Baum [1]. Experimental separation of strands with "good" codes that avoid intermolecular cross hybridization was reported in [4].

In [11], the authors introduce a theoretical approach to the problem of designing code words. Based on these ideas and code-theoretic properties, a computer program for generating code words is being developed [12]. Another algorithm based on backtracking, for generating such code words is also developed by Li [14]. Every biomolecular protocol involving DNA or RNA generates molecules whose sequences of nucleotides form a language over the four letter alphabet $\Delta = \{A, G, C, T\}$. The Watson-Crick (WC) complementarity of the nucleotides defines a natural involution mapping θ, $A \mapsto T$ and $G \mapsto C$ which is an anti-morphism of Δ^*. Undesirable WC bonds (undesirable hybridizations) can be avoided if the language satisfies certain coding properties. In particular for DNA code words, no involution of a word is a subword of another word, or no involution of a word is a subword of a composition of two words. These properties are called θ-compliance and θ-freedom respectively. The case when a DNA strand may form a hairpin, (i.e. when a word contains a reverse complement of a subword) was introduced in [12] and was called θ-subword compliance.

We start the paper with definitions of coding properties that avoid intermolecular and intramolecular cross hybridizations. The definitions of θ-compliant and θ-free languages are same as the ones introduced in [11]. Here we also consider intramolecular hybridizations and subword hybridizations. Hence, we have two additional coding properties: θ-subword compliance and θ-k-code. We make several observations about the closure properties of the code word languages. In particular, we concentrate on properties of languages that are preserved with union and concatenation. Assuming that two sets of DNA strands have "good" coding properties, if these properties are preserved under union, then by mixing the two sets of corresponding oligos in one tube, the coding properties are preserved. Also, if a set of DNA strands has "good" coding properties that are preserved under concatenation, then the same properties will be preserved under arbitrary ligation of the strands. Section 3 provides necessary and sufficient conditions for a finite set of words to generate (by concatenations) an infinite set of code words. In practice, besides use in ligation, these conditions provide a way to generate new "good" code words starting from a small set of initial "good" code words and as such might facilitate the otherwise difficult task of strand design. The section ends with several general methods of how to construct code words with certain desired properties. The last section provides conditions under which a splicing system generates a language of code words. These last observations extend results from [11]. We end with few concluding remarks. Due to page number constraints, we have omitted most of the proofs of our observations.

2 Definitions and Closure Properties

An alphabet Σ is a finite non-empty set of symbols. We will denote by Δ the special case when the alphabet is $\{A, G, C, T\}$ representing the DNA nucleotides.

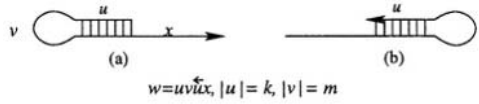


Fig. 1. Intramolecular hybridization (θ-subword compliance): (a) the reverse complement is at the beginning of the $5'$ end, (b) the reverse complement is at the end of the $3'$. The $3'$ end of the DNA strand is indicated with an arrow

A word u over Σ is a finite sequence of symbols in Σ. We denote by Σ^* the set of all words over Σ, including the empty word 1 and, by Σ^+, the set of all non-empty words over Σ. We note that with the word concatenation, Σ^* is the free monoid and Σ^+ is the free semigroup generated by Σ. The length of a word $u = a_1 \cdots a_n$ is n and is denoted with $|u|$.

For words representing DNA sequences we use the following convention. A word u over Δ denotes a DNA strand in its $5' \rightarrow 3'$ orientation. The Watson-Crick complement of the word u, also in orientation $5' \rightarrow 3'$ is denoted with $\overleftarrow{u}$. For example if $u = AGGC$ then $\overleftarrow{u} = GCCT$. There are two types of unwanted hybridizations: intramolecular and intermolecular. The intramolecular hybridization happens when two sequences, one being a reverse complement of the other appear within the same DNA strand (see Fig. 1). In this case the DNA strand forms a hairpin.

Two particular intermolecular hybridizations are of interest (see Fig. 2). In Fig. 2 (a) the strand labeled u is a reverse complement of a subsequence of the strand labeled v, and in the same figure (b) represents the case when u is the reverse complement of a portion of a concatenation of v and w.

Throughout the rest of the paper, we concentrate on finite sets $X \subseteq \Sigma^*$ that are *codes* i.e. every word in X^+ can be written uniquely as a product of words in X. For the background on codes we refer the reader to [3]. We will need the following definitions:

$$\begin{aligned} \mathrm{Pref}(w) &= \{u \mid \exists v \in \Sigma^*, uv = w\} \\ \mathrm{Suff}(w) &= \{u \mid \exists v \in \Sigma^*, vu = w\} \\ \mathrm{Sub}(w) &= \{u \mid \exists v_1, v_2 \in \Sigma^*, v_1 u v_2 = w\} \end{aligned}$$

Fig. 2. Two types of intermolecular hybridization: (a) (θ-compliant) one code word is a reverse complement of a subword of another code word, (b) (θ-free) a code word is a reverse complement of a subword of a concatenation of two other code words. The $3'$ end is indicated with an arrow

We define the set of prefixes, suffixes and subwords of a set of words. Similarly, we have $\mathrm{Suff}_k(w) = \mathrm{Suff}(w) \cap \Sigma^k$, $\mathrm{Pref}_k(w) = \mathrm{Pref}(w) \cap \Sigma^k$ and $\mathrm{Sub}_k(w) = \mathrm{Sub}(w) \cap \Sigma^k$.

We follow the definitions initiated in [11] and used in [12, 13].

An involution $\theta : \Sigma \to \Sigma$ of a set Σ is a mapping such that θ^2 equals the identity mapping, $\theta(\theta(x)) = x$, $\forall x \in \Sigma$.

The mapping $\nu : \Delta \to \Delta$ defined by $\nu(A) = T$, $\nu(T) = A$, $\nu(C) = G$, $\nu(G) = C$ is an involution on Δ and can be extended to a morphic involution of Δ^*. Since the Watson-Crick complementarity appears in a reverse orientation, we consider another involution $\rho : \Delta^* \to \Delta^*$ defined inductively, $\rho(s) = s$ for $s \in \Delta$ and $\rho(us) = \rho(s)\rho(u) = s\rho(u)$ for all $s \in \Delta$ and $u \in \Delta^*$. This involution is antimorphism such that $\rho(uv) = \rho(v)\rho(u)$. The Watson-Crick complementarity then is the antimorphic involution obtained with the composition $\nu\rho = \rho\nu$. Hence for a DNA strand u we have that $\rho\nu(u) = \nu\rho(u) = \overleftarrow{u}$. The involution ρ reverses the order of the letters in a word and as such is used in the rest of the paper.

For the general case, we concentrate on morphic and antimorphic involutions of Σ^* that we denote with θ. The notions of θ-free and θ-compliant in 2, 3 of Definition 1 below were initially introduced in [11]. Various other intermolecular possibilities for cross hybridizations were considered in [13] (see Fig. 3). All of these properties are included with θ-k-code (4 of Definition 1).

Definition 1. ***Let*** $\theta : \Sigma^* \to \Sigma^*$ *be a morphic or antimorphic involution. Let* $X \subseteq \Sigma^*$ *be a finite set.*

1. *The set* X *is called* $\theta(k, m_1, m_2)$-subword compliant *if for all* $u \in \Sigma^*$ *such that for all* $u \in \Sigma^k$ *we have* $\Sigma^* u \Sigma^m \theta(u) \Sigma^* \cap X = \emptyset$ *for* $m_1 \leq m \leq m_2$.
2. *We say that* X *is called* θ-compliant *if* $\Sigma^* \theta(X) \Sigma^+ \cap X = \emptyset$ *and* $\Sigma^+ \theta(X) \Sigma^* \cap X = \emptyset$.
3. *The set* X *is called* θ-free *if* $X^2 \cap \Sigma^+ \theta(X) \Sigma^+ = \emptyset$.
4. *The set* X *is called* θ-k-code *for some* $k > 0$ *if* $\mathrm{Sub}_k(X) \cap \mathrm{Sub}_k(\theta(X)) = \emptyset$.
5. *The set* X *is called* strictly θ *if* $X' \cap \theta(X') = \emptyset$ *where* $X' = X \setminus \{1\}$.

The notions of prefix, suffix (subword) compliance can be defined naturally from the notions described above, but since this paper does not investigate these properties separately, we don't list the formal definitions here.

We have the following observations:

Observation 1 In the following we assume that $k \leq \min\{ |x| : x \in X \}$.

1. X is strictly θ-compliant iff $\Sigma^* \theta(X) \Sigma^* \cap X = \emptyset$.
2. If X *is* strictly θ-free then X and $\theta(X)$ are strictly θ-compliant and $\theta(X)$ is θ-free.
3. If X is θ-k-code, then X is θ-k'-code for all $k' > k$.
4. X is a θ-k-code iff $\theta(X)$ is a θ-k-code.
5. If X is strictly θ such that X^2 is $\theta(k, 1, m)$-subword compliant, then X is strictly θ-k-code.

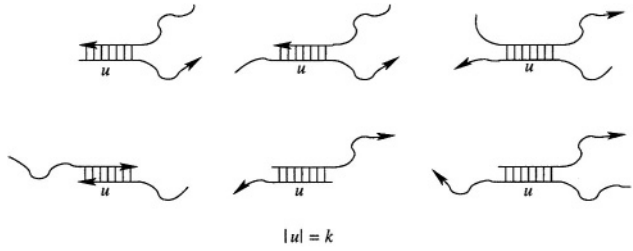


Fig. 3. Various cross hybridizations of molecules one of which contains subword of length k and the other its complement

6. If X is a θ-k-code then both X and $\theta(X)$ are $\theta(k,1,m)$-subword compliant, $\theta(k,1,m)$ prefix and suffix compliant for any $m \geq 1$, θ-compliant. If $k \leq \frac{|x|}{2}$ for all $x \in X$ then X is θ-free and hence avoids the cross hybridizations as shown in Fig. 1 and 2.
7. If X is a θ-k-code then X and $\theta(X)$ avoids all cross hybridizations of length k shown in Fig. 3 and so all cross hybridizations presented in Fig. 2 of [13].

It is clear that θ-subword compliance implies θ-prefix and θ-suffix compliance. We note that when $\theta = \rho\nu$, the $\theta(k, m_1, m_2)$-subword compliance of the code words $X \subseteq \Delta^*$ does not allow intramolecular hybridization as in Fig. 1 for a predetermined k and $m_1 \leq m \leq m_2$. The maximal length of a word that together with its reverse complement can appear as subwords of a code words is limited with k. The length of the hairpin, i.e. "distance" between the word and its reversed complement is bounded between m_1 and m_2. The values of k and m_1, m_2 would depend on the laboratory conditions (ex. the melting temperature and the length of the code words). In order to avoid intermolecular hybridizations as presented in Fig. 2, X has to satisfy θ-compliance and θ-freedom. Most applications would require X to be strictly θ. The most restricted and valuable properties are obtained with a θ-k-code, and the analysis of this type of codes is also most difficult. When X is θ-k-code, all intermolecular hybridizations presented in Fig. 3 are avoided. We include several observations in the next section.

2.1 Closure Properties

In this part of the paper we consider several closure properties of languages that satisfy some of the properties described with the Definition 1. We concentrate on closure properties of union and concatenation of "good" code words. From practical point of view, we would like to know under what conditions two sets of available "good" code words can be joined (union) such that the resulting set is still a "good" set of code words. Also, whenever ligation of strands is involved, we need to consider concatenation of code words. In this case it is useful to know under what conditions the "good" properties of the codewords will be preserved after arbitrary ligations. The following table shows closure properties of these languages under various operations.

	θ-subword compl.	θ-compl.	θ-free	θ-k-code
Union	Yes	No	No	No
Intersection	Yes	Yes	Yes	Yes
Complement	No	No	No	No
Concatenation ($XY, X \neq Y$)	No	Y/N	No	No
Kleene*	No	No	Yes	No

Table 1. Closure Properties

Most of the properties included in the table are straight forward. We add couple of notes for clarification.

- $\theta(k, m_1, m_2)$-subword compliant languages are not closed under concatenation and Kleene*. For example consider the set $X = \{aba^2\} \subseteq \{a, b\}^*$ with the morphic involution $\theta : a \mapsto b, b \mapsto a$. Then X is $\theta(2, 2, 4)$-subword compliant. But $X^2 = \{aba^3ba^2\}$ is not $\theta(2, 2, 4)$-subword compliant, i.e. X^2 contains $(ab)a^3\theta(ab)a$.
- When θ is a morphism θ-compliant languages are closed under concatenation, but when θ is antimorphism, they may not be. Consider the following example: $X_1 = \{a^2, ab\}$ and $X_2 = \{b^2, aba\}$ with the antimorphic involution $\theta : a \mapsto b, b \mapsto a$. Then $ab^3 \in X_1X_2$ and $bab^3 \in \theta(X_1X_2)$.

The next proposition which is a stronger version of Proposition 10 in [11], shows that for an antimorphic θ, concatenation of two distinct θ-compliant or θ-free languages is θ-compliant or θ-free whenever their union is θ-compliant i.e. θ-free respectively. Necessary and sufficient conditions under which a θ-compliant language is closed under Kleene* operation are considered in the next section.

Proposition 1. *Let $X, Y \subseteq \Sigma^+$ be two θ-compliant (θ-free) languages for a morphic or antimorphic θ. If $X \cup Y$ is θ-compliant (θ-free) then XY is θ-compliant (θ-free).*

3 Generating Infinite Sets of Code Words

By the results of the previous section we see that none of the θ-freedom, θ-compliance and θ-subword compliance are closed under arbitrary concatenation. In this section we investigate what are the properties of a finite set of "good" code words X that can generate an infinite set of code words X^* with the same "good" properties. In practice, it is much easier to generate a relatively small set of code words that has certain properties (i.e. in case of DNA or RNA, mismatched hybridization is avoided), and if we know that any concatenation of such words would also satisfy the requirements, the process of generating code words could be rather simplified. Hence, we give necessary and sufficient conditions for X such that X^* is θ-(subword) compliant or θ-free.

The Lemma below shows that if X is θ-free then X^2 is "almost" θ-compliant. The difference is in + vs $*$ in 2 of Definition 1.

Lemma 1. *If X is θ-free then $X^2 \cap \Sigma^+\theta(X^2)\Sigma^+ = \emptyset$.*

Note that the converse of the above need not be true. For example consider $X = \{a^3b, a^2b^2\}$ with θ being morphism $a \mapsto b, b \mapsto a$. Then $X^2 \cap \Sigma^+\theta(X^2)\Sigma^+ = \emptyset$ since all words in X^2 are of length 8, but X is not θ-free since $a^2b^2a^3b = a^2\theta(a^2b^2)ab$.

Lemma 2. *If X is strictly θ-compliant then X^n is strictly θ-compliant for all $n > 1$.*

The Kleene * closure of X contains the union of all X^n and in order for it to be θ-compliant we need stronger properties. The next proposition is a stronger version of Lemma 1 (ii) and Proposition 1 in [11].

Proposition 2. *The follwoing are equivalent:*

(i) X is strictly θ-free
(ii) X^* is strictly θ-compliant
(iii) X^* is strictly θ-free

The following proposition investigates the case when the property of subword compliance is preserved with Kleene $*$. It turned out that conditions under which a subword compliant set of code words is closed under concatenation with itself are somewhat more demanding than the ones for θ-free and θ-compliant. Considering 5 in Observation 1 these properties might turn out to be quite important.

Proposition 3. *Let k, m_1, m_2 and $m_1 \leq m \leq m_2$ be positive integers and $X \subseteq \Sigma^*$ be such that for every word $x \in X$, $|x| \geq 2k+m_2$. Let $L = \cup \mathrm{Suff}_i(X)\mathrm{Pref}_j(X)$ where the union is taken such that $i + j = 2k + m$ for $0 \leq i, j \leq 2k + m$.*

Then X^* is $\theta(k, m_1, m_2)$-subword compliant if and only if X is $\theta(k, m_1, m_2)$-subword compliant and for all $y \in L$ $\mathrm{Pref}_k(y) \cap \mathrm{Suff}_k(\theta(y)) = \emptyset$.

The conditions under which codes with one of the coding properties in Definition 1 are closed under Kleene* are discussed above. But when is a language $\theta(k, m_1, m_2)$-subword compliant, θ-compliant and θ-free all at once? When can we generate infinite set of code words such that the set avoids all kinds of hybridizations as shown in Fig. 3. The next two propositions try to answer some of these questions. The condition in the first proposition is quite strong due to the strong requirements. However, the condition is only sufficient, and may not be necessary.

Proposition 4. *Let X be a θ-2-code such that $\forall x \in X$, $|x| \geq 3$. Then both X and X^* are strictly*

1. *$\theta(k, m_1, m_2)$-subword compliant for $k \geq 3$ for all $m_1 < m_2$.*
2. *θ-compliant and θ-free.*

The following observation is straight forward.

Proposition 5. *Let X be a θ-k-code for $k \leq \min\{|x| : x \in X\}$. Then X^* is strictly θ-k-code iff X^2 is a strictly θ-k-code.*

3.1 Methods to Generate Good Code Words

With previous section we describe the necessary and sufficient conditions under which concatenations of "good" code words produce new "good" code words. With the following few observations we show several ways to generate such codes. Many authors have realized that in the design of DNA strands it is helpful to consider three out of the four bases. This was the case with several successful experiments [2, 6, 15]. It turns out that this, or a variation of this technique can be generalized such that codes with some of the desired properties can be easily constructed. The first proposition below shows how starting from one word w that satisfies certain properties we can construct strictly θ-free set of codes.

Proposition 6. *Given an alphabet set Σ and an involution θ such that $|\Sigma| \geq 2$ and θ not identity. Consider a word $w \in \Sigma^+$, $|w| \geq 3$ such that* $\mathrm{Sub}(w)$ *is strictly θ. Let, $P = \bigcup_{2\leq i\leq |w|-1} \mathrm{Pref}_i(\theta(w))$ and $S = \bigcup_{2\leq i\leq |w|-1} \mathrm{Suff}_i(\theta(w))$ such that $P^2 \cap (P \cup S) = \emptyset$.*

Then for any set $X \subseteq \cup_{2\leq i\leq |w|-1} P\mathrm{Pref}_i(w)P$ we have that both X and X^* are strictly θ-free.

Example 2. Consider the DNA alphabet Δ. Let w =CAACA and $\rho\nu(w)$ = TGTTG. Hence P = {TG, TGT,TGTT }, S = {TG, TTG, GTTG} and $\cup_{2\leq i\leq 4}\mathrm{Pref}_i(w) = \{CA, CAA, CAAC\}$. Note that concatenation of any two words in P is distinct from the words in $P \cup S$. Hence $X \subseteq \bigcup_{2\leq i\leq 4} P\mathrm{Pref}_i(w)P$ is such that both X and X^* are strictly $\theta = \rho\nu$-free.

The following proposition is a general version of Proposition 16 of [11].

Proposition 7. *Given an alphabet set Σ and an involution θ such that $|\Sigma| \geq 3$ and θ not identity. Choose distinct $a, b, c \in \Sigma$ such that $\theta(a) = b, c$. Let $X = \bigcup_{i=1}^{\infty} a^m(\Sigma^{m-1}c)^i b^m$ for some $m \geq 2$. Then X, and so X^* is strictly θ-free.*

Example 3. Consider Δ and $\theta = \rho\nu$ and let $m = 2$, $a = A, c = C, b = G$.
Then $X = \bigcup_{i=1}^{\infty} AA(\Delta C)^i GG$ and X^* are strictly θ-free.

With the following propositions we consider ways to generate θ-subword compliant and θ-k-codes.

Proposition 8. *Given an alphabet set Σ such that $|\Sigma| \geq 3$ and an involution (morphic or antimorphic) θ such that θ is not identity. Let $a, b \in \Sigma$ such that $\theta(a) = b$ and let $X = \bigcup_{i=1}^{\infty} a^{k-1}((\Sigma \setminus \{a, b\})^{k-2}b)^i$. Then X is $\theta(k, 1, m)$-subword compliant for $k \geq 3$ and $m \geq 1$. Moreover, when θ is morphic then X is a θ-k-code.*

Example 4. Consider Δ with $\theta = \rho\nu$ and choose $k = 3$.
Then $X = \bigcup_{i=1}^{\infty} AA(\{G, C\}T)^i$ is $\theta(3, 1, m)$-subword compliant for any $m \geq 1$.

As other authors have observed, note that it is easy to get θ-k-code if one of the symbols in the alphabet is completely ignored in the construction of the code X.

Proposition 9. *Given an alphabet set Σ such that $|\Sigma| \geq 3$ and an involution (morphic or antimorphic) θ such that θ is not identity. Let $b, c \in \Sigma$ such that $\theta(b) = c$ and let $X = \bigcup_{i=1}^{\infty} a^{k-1}((\Sigma \setminus \{c\})^{k-2}b)^i$. Then X is θ-k-code.*

4 Codes for Splicing

Splicing systems were introduced in [9] as a model for the cut and paste activity made possible through the action of restriction enzymes and a ligase on double stranded DNA molecules. They were further developed in computational models (see for example [10, 17]). In this section we consider the question of determining the properties of the code words such that under splicing they produce new "good" code words, in other words, during the computational process the "good" encoding is not lost.

First we define a notion of a splicing base which is more general than the one defined in [11] (called strong splicing base here). We show that the splicing rules that are drawn from the splicing base preserve the θ-compliance and under some additional properties, they preserve θ-freedom and θ-subword compliance.

Notation: in what follows, for a set B we denote with $B^{(4)}$ the direct product $B \times B \times B \times B$.

A *splicing system* is an ordered triple $\gamma = (\Sigma, A, \mathcal{R})$ where Σ is a finite alphabet, $A \subset \Sigma^*$ is a set of words called *axioms* and $\mathcal{R} \subseteq (\Sigma^*)^{(4)}$ is a set of splicing rules. The splicing operation σ is defined such that if $r = (u_1, u_2, v_1, v_2)$ then

$$\begin{matrix} x = x_1u_1u_2x_2 \\ y = y_1v_1v_2y_2 \end{matrix} \text{ rule } r \text{ produces } (\Rightarrow_r) \begin{matrix} w_1 = x_1u_1v_2y_2 \\ w_2 = y_1v_1u_2x_2 \end{matrix}$$

and in this case we write $(x, y) \Rightarrow_r (w_1, w_2)$. For a language $L \subseteq \Sigma^*$ and a set of rules $\mathcal{R}$ we say that $\sigma(L)$ is obtained by single step splicing of L if $\sigma(L) = \{w \mid \exists r \in \mathcal{R}, \exists x, y \in L, (x, y) \Rightarrow_r (w, w')\}$. Then the *language generated by the splicing system* γ with axiom set A is $L(\gamma) = \cup_{n \geq 0} \sigma^n(A)$.

For the background on splicing systems we refer the reader to [10, 17].

For a morphic or antimorphic involution θ we define special θ-rules that preserve the "good" codes.

4.1 θ-rules:

In the following X is a finite code $\subseteq \Sigma^+$ and θ is an antimorphic or morphic involution, $\theta : \Sigma^* \to \Sigma^*$.

Definition 2. *Let X be a set and $B \subset \mathrm{Sub}(X)$ be a subset of words with $k = \min\{|x| : x \in B\} \geq 0$. We say that a set B is a* base for splicing rules for X *if it is strictly θ and satisfies the following two properties:*

1. $\mathrm{Sub}_s(\theta(B^2)) \cap \mathrm{Sub}_s(B^2) = \emptyset$ *for all* $s \geq k$
2. $\theta(B^2) \cap \mathrm{Sub}(X) = \emptyset$.

Definition 3. *Let X be a set of words and B a base for splicing rules for X. Then a set $\mathcal{R}_B(X)$ which is a subset of $B^{(4)}$ is called a set of* θ-rules *for X.*

4.2 Several Observations About θ-rules

1. The θ-rules can always be reflexive. This means for each rule (u_1, u_2, v_1, v_2) in $\mathcal{R}_B(X)$ the rules (u_1, u_2, u_1, u_2) and (v_1, v_2, v_1, v_2) can also be in $\mathcal{R}_B(X)$. This follows directly from the definition of θ-rules.
2. θ-rules can be symmetric. If $(u_1, u_2, u_3, u_4) \in \mathcal{R}_B(X)$ then $(u_3, u_4, u_1, u_2) \in \mathcal{R}_B(X)$. This also follows from the definition of θ-rules.
3. One way to obtain a set of θ-rules for X is the following. Let $m = \min\{|x| : x \in X\}$ and $k = \lceil \frac{m}{2} \rceil$.
   Note that even if X is strictly θ, $\mathrm{Sub}_k(X)$ may not necessarily be so. We partition $\mathrm{Sub}_k(X)$ into three strictly θ subsets in the following way. Set $U_1 = \{x \mid x \notin \mathrm{Sub}_k(\theta(X))\}$ and $U_2 = \mathrm{Sub}_k(X) \setminus U_1$. Then by definition U_1 is strictly θ and for every $u \in U_2$ we have $\theta(u) \in U_2$. Now we partition U_2 into U_2' and U_2'' such that $u \in U_2'$ and $\theta(u) \in U_2''$. Hence $\mathrm{Sub}_k(X) = U_1 \cup U_2' \cup U_2''$ is a partition of $\mathrm{Sub}_k(X)$ into three strictly θ sets. We form the basis of the words used in the splicing rules from two of these sets.
   Let W be either U_2' or U_2'' and let $B = U_1 \cup W$. Then B is a maximal set of subwords of X of length k that is strictly θ. Now we remove from B all words that violate the properties (1) and (2) of Definition 2. This may leave an empty set of B. In that case we consider the subwords of length $k-1$ and $k+1$ and repeat the procedure for obtaining B. This procedure ends either with a base B for splicing rules or, since X *is* finite, with no base for splicing rules for X.
4. A set K is called *strong splicing base* if for all $x \in \Sigma^*$ and for all $u \in K$, if xu is a prefix of K^* then $x \in K^*$ and if ux is a suffix of K^* then $x \in K^*$. In [11] the strong splicing base is called simply a splicing base. We have to make a distinction in our case since the base for splicing rules is not necessarily a strong splicing base (see Example 5). However, the base for splicing rules B obtained from the construction in 3 above, with words of constant length, does give us a strong splicing base.

The need for the set of θ-rules $\mathcal{R}$ to be reflexive and symmetric comes naturally from the chemistry of the restriction enzymes. If an enzyme can cut one molecule, the ligase can recombine the same molecule back together, hence a reflexive operation. If two molecules x and y take part in a splicing operation (can be cut by an enzyme) then the same molecules written in the opposite order y and x are part of the same operation. Hence the symmetric operation. In the following we assume that the splicing rules are reflexive and symmetric.

Proposition 10. *Let $\gamma = (\Sigma, A, \mathcal{R}_B(A))$, be a splicing system with $\mathcal{R}_B(A)$ being the a set of θ-rules. If the axiom set A is θ-compliant then the language generated by the system $L(\gamma)$ is strictly θ-compliant.*

The following proposition provides conditions under which the language generated by a splicing system is θ-free.

Proposition 11. *Let $\gamma = (\Sigma, A, \mathcal{R}_B(A))$ be a splicing system $\mathcal{R}_B(A)$ being a set of θ-rules for A. Assume that the base B for the splicing rules is such that*

$$\{\theta(B^2)\} \cap \mathrm{Sub}(A^2) = \emptyset$$

If the axiom set A is strictly θ-free, then $L(\gamma)$ is also θ-free.

We would like to point out that in [11] it was proven that if a strong splicing base is strictly θ-free, then so is the language generated by the splicing system, provided that the set of axioms is a subset of the free monoid generated by the splicing base. The Proposition 11 does not require that the axiom set is a subset of the free monoid generated by the base for the splicing rules, and therefore, the language generated by the splicing system is not necesarily a subset of this same monoid (as is the case in [11]). The following example shows a base for splicing rules that is not strong splicing base.

Example 5. Consider the alphabet $\Sigma = \{a, b, c\}$ with the morphic involution θ defined with $a \mapsto c$, $b \mapsto b$ and $c \mapsto a$. Let the axiom set $A = \{abaaba\}$. We choose a base for splicing rules to be $B = \{a, ab\}$. This is clearly a strictly θ-free code, but it is not a strong splicing base. Consider $a \cdot ab \cdot a \in B^+$ with suffix aba. If we pick $u = a$ and $x = ba$ we have that ux is a suffix of a word in B^+ and $u \in B$ but $x \notin B^+$. Now, $\theta(B^2) = \{cc, ccb, cbc, cbcb\}$ and clearly the conditions of the definition 2 and Proposition 11 are satisfied. We can take $\mathcal{R} = B^{(4)}$ and the resulting generated splicing language is infinite (contains $aba^+ba \cup (ab)^+a$) and is strictly θ-free.

The following proposition provides conditions under which the language generated by a splicing system is θ-k-code.

Proposition 12. *Let $\gamma = (\Sigma, A, \mathcal{R}_B(A))$ be a splicing system $\mathcal{R}_B(A)$ being a set of θ-rules for A. Assume that the base B for the splicing rules is such that*

$$\mathrm{Sub}_k\theta(B^2) \cap \mathrm{Sub}_k(A) = \emptyset$$

If the axiom set A is θ-k-code, then $L(\gamma)$ is also θ-k-code.

We end the paper with an observation about the conditions under which a splicing language is θ-subword compliant.

Proposition 13. *Given k, let A be such that $\mathrm{Sub}_k(A) \cap \mathrm{Sub}_k(\theta(A)) = \emptyset$. Let $\mathcal{R}_B(A) \subseteq B^{(4)}$ be a set of rules with words from a set $B \subseteq \mathrm{Sub}_k(A)$ that satisfies $\theta(\mathrm{Suff}_i(B)\mathrm{Pref}_j(B)) \cap \mathrm{Sub}_k(A) = \emptyset$ and $\theta(\mathrm{Suff}_i(B)\mathrm{Pref}_j(B)) \cap \mathrm{Suff}_r(B)\mathrm{Pref}_s(B) = \emptyset$ for all $i+j = k$ and $r+s = k$. Then $L(\gamma)$ is $\theta(k, m_1, m_2)$-subword compliant for all $m_1 < m_2$.*

Example 6. $X = \{(ab)^n : 1 \leq n \leq 10\}$ is θ-free and $\theta(2, 2, 5)$-subword compliant for $\Sigma = \{a, b, c, d\}$ with the morphic involution θ defined with $a \mapsto c$ and $b \mapsto d$. Then $\theta(X) = \{(cd)^n : 1 \leq n \leq 10\}$ and consider the set $U = \{a, b\}$ as is defined in 3 of the Subsection 4.2. It is easy to verify that $L(\gamma)$ for the splicing system defined in Subsection 4.2(3) is infinite and $\theta(2, 2, 5)$-subword compliant, θ-compliant and θ-free.

5 Concluding Remarks

In this paper we investigated theoretical properties of languages that consist of DNA based code words. In particular we concentrated on intermolecular and intramolecular cross hybridizations that can occur as a result that a Watson-Crick complement of a (sub)word of a code word is also a (sub)word of a code word. These conditions are necessary for a design of good codes, but certainly may not be sufficient. For example, the algorithms used in the programs developed by Seeman [19], Feldkamp [7] and Ruben [18], all check for uniqueness of k-length subsequences in the code words. Unfortunately, none of the properties from Definition 1 ensures uniqueness of k-length words. Such code word properties reamin to be investigated. The observations in Section 3 provide a general way how from a small set of code words with desired property we can obtain, by concatenating the existing words, arbitrarily large sets of code words with similar properties. We hope that the general methods of designing such codewords will simplify the search for "good" codes. Better characterizations of good code words that are closed under Kleene * operation may provide even faster ways for designing such codewords. The most challenging questions of characterizing and designing good θ-k-codes remains to be developed.

Our approach to the question of designing "good" DNA codes has been from the formal language theory aspect. Many issues that are involved in designing such codes have not been considered. These include (and are not limited to) the free energy conditions, melting temperature as well as Hamming distance conditions. All these remain to be challenging problems and a procedure that includes all or majority of these aspects will be desirable in practice. It may be the case that regardless of the way the codes are designed, the ultimate test for the "goodness" of the codes will be in the laboratory.

Acknowledgement

This work has been partially supported by grants EIA-0086015 and EIA-0074808 from the National Science Foundation, USA.

References

[1] E. B. Baum, *DNA Sequences useful for computation* unpublished article, available at: http://www.neci.nj.nec.com/homepages/eric/seq.ps (1996).

[2] R. S. Braich, N. Chelyapov, C. Johnson, P. W. K. Rothemund, L. Adleman, *Solution of a 20-variable 3-SAT problem on a DNA computer,* Science **296** (2002) 499-502.

[3] J. Berstel, D. Perrin, *Theory of codes,* Academis Press, Inc. Orlando Florida, 1985.

[4] R. Deaton, J. Chen, H. Bi, M. Garzon, H. Rubin, D.F. Wood, *A PCR-based protocol for in vitro selection of non-crosshybridizing oligonucleotides, DNA Computing: Proceedings of the 8th International Meeting on DNA Based Computers* (M. Hagiya, A. Ohuchi editors), Springer LNCS **2568** (2003) 196-204.

[5] R. Deaton et. al, *A DNA based implementation of an evolutionary search for good encodings for DNA computation, Proc. IEEE Conference on Evolutionary Computation* ICEC-97 (1997) 267-271.
[6] D. Faulhammer, A. R. Cukras, R. J. Lipton, L. F.Landweber, *Molecular Computation: RNA solutions to chess problems,* Proceedings of the National Academy of Sciences, USA **97** 4 (2000) 1385-1389.
[7] U. Feldkamp, S. Saghafi, H. Rauhe, *DNASequenceGenerator - A program for the construction of DNA sequences, DNA Computing: Proceedings of the 7th International Meeting on DNA Based Computers* (N. Jonoska, N. C. Seeman editors), Springer LNCS **2340** (2002) 23-32.
[8] M. Garzon, R. Deaton, D. Reanult, *Virtual test tubes: a new methodology for computing, Proc. 7th. Int. Symposium on String Processing and Information retrieval, A Coruña, Spain.* IEEE Computing Society Press (2000) 116-121.
[9] T. Head, *Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, Bull. Math. Biology* **49** (1987) 737-759.
[10] T. Head, Gh. Paun, D. Pixton, *Language theory and molecular genetics, Handbook of formal languages, Vol.II* (G. Rozenberg, A. Salomaa editors) Springer Verlag (1997) 295-358.
[11] S. Hussini, L. Kari, S. Konstantinidis, *Coding properties of DNA languages, DNA Computing: Proceedings of the 7th International Meeting on DNA Based Computers* (N. Jonoska, N.C. Seeman editors), Springer LNCS **2340** (2002) 57-69.
[12] N. Jonoska, D. Kephart, K. Mahalingam, *Generating DNA code words* Congressus Numernatium **156** (2002) 99-110.
[13] L. Kari, S. Konstantinidis, E. Losseva, G. Wozniak, *Sticky-free and overhang-free DNA languages* preprint.
[14] Z. Li, *Construct DNA code words using backtrack algorithm,* preprint.
[15] Q. Liu et al., *DNA computing on surfaces,* Nature **403** (2000) 175-179.
[16] A. Marathe, A. E. Condon, R. M. Corn, *On combinatorial word design,* Preliminary Preproceedings of the 5th International Meeting on DNA Based Computers, Boston (1999) 75-88.
[17] Gh. Paun, G. Rozenberg, A. Salomaa, *DNA Computing, new computing paradigms,* Springer Verlag 1998.
[18] A. J. Ruben, S. J. Freeland, L. F. Landweber, *PUNCH: An evolutionary algorithm for optimizing bit set selection, DNA Computing: Proceedings of the 7th International Meeting on DNA Based Computers* (N. Jonoska, N.C. Seeman editors), Springer LNCS **2340** (2002) 150-160.
[19] N. C. Seeman, *De Novo design of sequences for nucleic acid structural engineering* J. of Biomolecular Structure & Dynamics **8** (3) (1990) 573-581.

Secondary Structure Design of Multi-state DNA Machines Based on Sequential Structure Transitions

Hiroki Uejima and Masami Hagiya

Japan Science and Technology Corporation (JST-CREST) and
Department of Computer Science,
Graduate School of Information Science and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
{uejima, hagiya}@is.s.u-tokyo.ac.jp

Abstract. This paper deals with the problem of designing the secondary structure of a multi-state molecular machine in which the formation of repeated DNA hairpin structures changes sequentially with the aim of implementing more sophisticated DNA nanomachines. Existing methods are insufficient to construct such a huge molecular machine using multiple DNA molecules. The method used in this paper validates the changes in formation exhaustively by dividing the secondary structure into hairpin units. It considers the minimum free energy of the structure, the structure transition paths, and the total frequency of optimal and sub-optimal structures. Hence, it can better design base sequences using the principles of thermodynamics.

1 Introduction

1.1 Multi-State Molecular Machine

In the field of DNA nanotechnology, various nanomachines made of DNA molecules have been implemented. Mao *et al.* [9] proposed a DNA motor based on changes in the structure of the DNA helix (B and Z) with salt concentration. However, changes in salt concentration affect all the DNA molecules in a solution uniformly, and each motor cannot be operated individually regardless of its base sequence. Moreover, the motor is restricted to two states. Yurke *et al.* [20] proposed a molecular system called the molecular tweezers, which has two states that result from changes in its secondary DNA structure. This system depends on the base sequences of its DNA molecules and can be operated individually. Simmel and Yurke then implemented a three-state machine [15] by extending their nanoactuator [14]. Unfortunately, it is not obvious how to extend this to general multi-state machines. Yan *et al.* applied 'fuelling', which was established in the aforementioned works to control the transition between $\mathtt{JX}_2$ and PX DNA tiles [19].

The basis for constructing more generalized nanoscale machines and computing devices is to implement multi-state machines using molecules. This study

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 74–85, 2004.

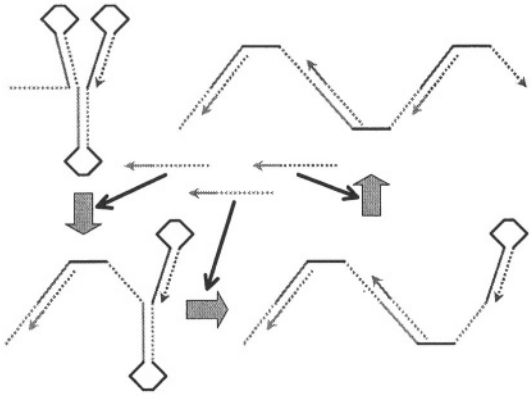


Fig. 1. Repeated hairpin structures in DNA molecules are opened by hybridizing the structures with DNA oligomers in a specific order. Here, the solid and dashed lines of the same color are complementary sequences

examines the design of a multi-state molecular machine that undergoes sequential state transitions as a result of multiple inputs.

Prototype Hairpin-Based State Machine A conformational state machine is a multi-state machine in which its conformation (i.e., secondary structure) indicates its state. We propose a hairpin-based state machine that uses the sequential opening of DNA hairpins to implement a conformational state machine. This system consists of DNA hairpins and oligomers whose sequences appear in the hairpin stems. A DNA oligomer can open the corresponding hairpin structure by invading the hairpin stem via branch migration. The hairpins are concatenated with an additional sticky end and form repeated hairpin structures in a single DNA strand. The entire series of repeated hairpin structures comprises a multi-state machine, which maintains its state with its hairpin structures, and the DNA oligomers function as state transition signals.

Initially, an oligomer can interact with the hairpin structure at the end of a strand with a sticky end. If the sequences of the hairpin stem and oligomer match, the oligomer invades the hairpin structure via branch migration after hybridizing with the sticky end. When the hairpin opens, a new sticky end is revealed and it plays a role in opening the next hairpin. Consequently, the hairpin structures are opened sequentially by the corresponding oligomers starting from one end of the DNA strand, as depicted in Figure 1.

Following Turberfield *et al.* [16], who pointed out that bulge loops can inhibit hybridization, a similar state machine can be implemented using bulge loops, as shown in Figure 2. This machine is superior to ours in that the topological configuration of the strands more strongly inhibits the invasion of a bulge loop by an oligomer. Moreover, it is also inhibited by the stiffness of the double strand that is formed. Rather than inhibiting hybridization, our machine uses a hairpin

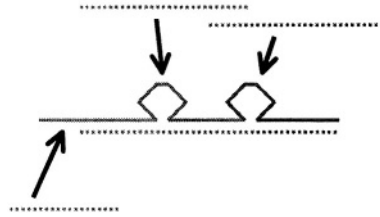


Fig. 2. The repeated bulge structures of DNA molecules are opened by hybridizing them with DNA oligomers in a specific order

to reveal the next single-stranded part after it is opened by an oligomer. As our machine is simpler, because it is made of a single strand, if it is proven to work robustly, it can be used as another type of building block for DNA machines.

1.2 Thermodynamic Analysis of DNA Hybridization

Thermodynamic models, such as nearest-neighbor (NN) thermodynamics, can be used to analyze secondary structures quantitatively. NN thermodynamics considers the stability of the secondary structure. It assumes that the stability of a given base pair depends on the identity and orientation of the neighboring base pairs. In this study, we use the thermodynamic parameters reported by John SantaLucia Jr. [13] for the NN model.

The folding problem calculates the secondary structure into which a given base sequence folds to give the most stable structure. Zuker *et al.* [21] proposed a dynamic programming algorithm to solve this problem in the polynomial time $O(n^3)$, where n is the length of the base sequence. Their algorithm is one of the algorithms implemented in "the Vienna RNA Package" by Hofacker *et al.* [7] and in "mfold" by Zuker *et al.*

The inverse folding problem calculates the base sequence that folds into a given secondary structure as the most stable structure. The algorithm is based on a simple search that is evaluated using a folding function that computes the similarity between the target structure and the minimum free energy structure of a sequence. It is also implemented in "the Vienna RNA Package" by Hofacker *et al.*

The distribution of secondary structures formed by a DNA/RNA molecule in the equilibrium state depends on the partition function of the sequence and the free energy of each structure. Therefore, the minimum free energy structure by itself is not sufficient to predict the actual behavior of a molecule. Wuchty *et al.* [18] proposed an algorithm that finds the complete set of sub-optimal RNA structures. Their algorithm was a modification of the algorithm used to find the optimal structure. This algorithm was also implemented by Hofacker *et al.*

As mentioned above, the partition function is one of the most important factors for predicting the behavior of DNA/RNA molecules. McCaskill [10] solved this problem by using programming in a manner similar to the folding problem.

The time complexity of his algorithm is $O(n^3)$. The algorithm is also implemented in "the Vienna RNA Package" by Hofacker *et al.*

There are two tractable ways to approximate the energy barrier height between two given structures. One method generates transition paths randomly and finds their lowest energy mountaintop. A transition path is generated in such a way that the transition proceeds more frequently in the direction with the smaller increase in the free energy. This algorithm was proposed by Flamm *et al.* [6] The other takes advantage of the heuristic proposed by Morgan *et al.* [11] and is based on a simplified energy model of secondary structures. Their model uses the number of base pairs as the free energy of a structure. Their heuristic generates a path with a very low energy mountaintop efficiently, but guarantees nothing about the properties of the path.

2 Method

First, we introduce the criteria of selectivity and ordinality, which need to be satisfied by our hairpin-based state machine. Then, we explain what frequencies of structures should be focused on to design the molecular machine.

2.1 Formalism

The selectivity and ordinality should be guaranteed by any number of hairpin sequences concatenated in any order. These criteria are reduced to conditions involving a minimum of two repeated hairpin structures, because successive hairpin opening is an orderly behavior. Only the combination of a sticky end sequence and a hairpin sequence needs to be verified to guarantee selectivity. As for ordinality, only the combination of two hairpin sequences needs to be verified.

Selectivity We call the oligomer that opens a hairpin structure an *input oligomer* (Figure 3 (a), (b)). The input oligomer consists of two parts. The part that hybridizes with the sticky end of a hairpin is called the *head* (the green part in Figure 3 (a)), and the part that invades and hybridizes with the stem of the hairpin is called the *tail* (the red part in Figure 3 (a)). The sticky end of a hairpin is also part of the stem of another hairpin.

Some of the notation used for sequences is defined here. $\mathrm{Hairpin}(s_1)$ is the sequence of the hairpin structure that includes the sequences s_1 and $\overline{s_1}$ in its stem, where $\overline{s_1}$ denotes the complementary sequence of s_1. $\mathrm{Sticky}(s_1)$ is the sequence of the stem part of $\mathrm{Hairpin}(s_1)$, which functions as the sticky end. Therefore, if hairpins are opened from the 5'-end, $\mathrm{Sticky}(s_1) = \overline{s_1}$. $\mathrm{Sticky}(s_1)\mathrm{Hairpin}(s_2)$ indicates that the sequence of the hairpin structure is concatenated at its 5'-end with the sequence of the sticky end. For example, the sequence of the structure shown in Figure 3 (1) is represented by $\mathrm{Sticky}(s_1)\mathrm{Hairpin}(s_2)$, where the sequence $\overline{s_1}$ corresponds to the green dotted line, and s_2 and $\overline{s_2}$ correspond to the solid and dotted red lines, respectively.

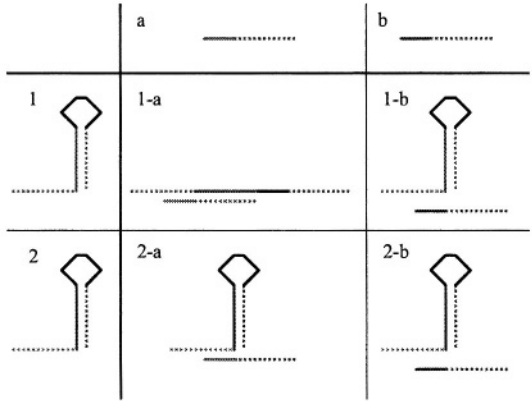


Fig. 3. A schematic of the requirements for selectivity

Opener(s_1, s_2) is the opener sequence that consists of part of Hairpin(s_1) as its head and part of Hairpin(s_2) as its tail. Therefore, if hairpins are opened from the 5'-end, Opener(s_1, s_2) is $\overline{s_2}s_1$ or a part of $\overline{s_2}s_1$. For example, the oligomer shown in Figure 3 (a) is represented by Opener(s_1, s_2), where part of sequence s_1 corresponds to the green solid line, and $\overline{s_2}$ corresponds to the red dotted line.

The selectivity of the set of hairpin sequences Hairpin(x) $(x = s_1, \ldots, s_n)$ is defined as follows: The hairpin structure Sticky(x)Hairpin(y) is opened by the oligomer Opener(x, y) for any $x, y \in \{s_1, \ldots, s_n\}$ $(x \neq y)$. In this case, the hairpin structures are opened properly (Figure 3 (1-a)). The hairpin structure Sticky(x)Hairpin(y) is not opened by the oligomer Opener(z, w), where $x \neq z$ or $y \neq w$, for any $x, y, z, w \in \{s_1, \ldots, s_n\}$ $(x \neq y, z \neq w)$. In this case, the hairpin structures are never opened (Figure 3 (1-b), (2-a), (2-b)).

In case (1-b), the oligomer may invade the hairpin structure without hybridizing with the sticky end and open the hairpin. If we identify the oligomer using its tail sequence, such a situation causes no problem, because the tail of the oligomer agrees with the hairpin. In other words, both (a) and (b) are identical signals for opening the red hairpin structure regardless of the sticky end. Hence, this case is omitted when confirming the selectivity. An invading oligomer without a sticky end is involved in ordinality rather than in selectivity.

Ordinality Similarly, the ordinality of the set of hairpin sequences Hairpin(x) $(x = s_1, \ldots, s_n)$ is defined as follows: The two sequential hairpin structures Hairpin(x)Hairpin(y) are not opened by the oligomer Opener(z, w) for any $x, y, z, w \in \{s_1, \ldots, s_n\}$ $(x \neq y, z \neq w)$. Namely, a hairpin should not be opened until the adjacent hairpin is opened.

The cases in which the hairpin and tail do not agree are not verified because the hairpin is seldom opened in such cases. Therefore, only the cases shown in Figure 4 are verified.

Fig. 4. A schematic of the requirements for ordinality

2.2 Procedure

First, we describe how selectivity and ordinality are verified based on these definitions. Next, we explain how to calculate the frequencies of the structures corresponding to these criteria. We implemented this procedure using the C programming language with the library of the Vienna RNA package.

Verifying Selectivity Selectivity can be confirmed using a simple condition:

> Given the strands of a hairpin structure and an oligomer, their minimum free energy structure is *similar to* the target structure (the opened or closed hairpin structure).

This condition is verified as an instance of the folding problem. To use Zuker's folding algorithm [21], these two strands are concatenated with virtual bases that cannot hybridize with any base and they are dealt with as one strand.

A secondary structure can be considered as a set of base pairs. Naturally, the similarity or distance between two structures is defined by the size of the symmetric difference between the sets corresponding to the structures. In short, the distance $d(\Omega_1, \Omega_2)$ between secondary structures Ω_1 and Ω_2 is:

$$d(\Omega_1, \Omega_2) = |\Omega_1 \Delta \Omega_2| = |(\Omega_1 \cup \Omega_2) \setminus (\Omega_1 \cap \Omega_2)|.$$

When the target structure is Ω, the structure Ω' such that $d(\Omega, \Omega') \leq D$ is *similar* to Ω and these two structures can be identified. The threshold D is based on the target structure and size.

Verifying Ordinality Selectivity can be confirmed by folding all combinations of sequences as explained in the previous section and checking that the target structure is similar to the optimal one. However, it is impossible to satisfy ordinality in this way. For some combinations of sequences, the DNA oligomer hybridizing with and opening the hairpin structure is the minimum free energy structure, even when the sticky end of the hairpin is not included. Although a structure violating ordinality might be the minimum free energy structure, the high-energy barrier on the transition path leading to the violating structure guarantees the rarity of violations of ordinality in actual situations. Minimizing the valley depth also makes the violating structure less stable. Therefore, our

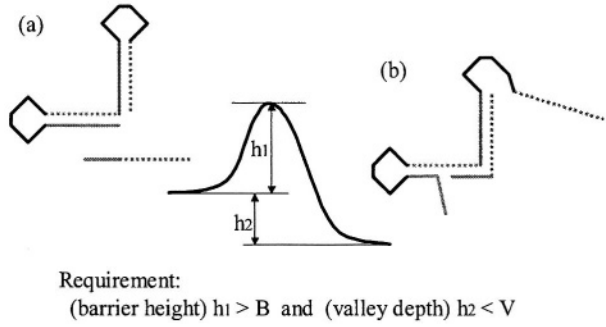


Fig. 5. An energy curve of a structure transition violating ordinality

program checks whether the energy barrier is higher than a given threshold and the energy valley is shallower than a second given threshold.

The depth of the energy valley is the difference between the energies of the initial (Figure 5 (a)) and final (Figure 5 (b)) structures. The latter is the hairpin opened by an improper invasion of the oligomer. The barrier height is the lowest energy peak in the structure transition. In our program, the barrier height [17] is approximated using the lowest peak for several paths generated with Morgan and Higgs' algorithm [11].

However, justifying and improving this condition for ordinality requires more precise analyses of hairpin opening, including kinetic analyses. This is left for a future study and is briefly discussed in the final section.

Maximizing the Frequency of a Structure In addition to requiring that the target structure be similar to the minimum free energy structure, the frequency of the target structure should also be maximized. This frequency can be computed as the sum of the frequencies of structures similar to the target. For example, the frequency $\tilde{F}_{\text{selectivity}}$ used to verify the selectivity is:

$$\tilde{F}_{\text{selectivity}} = \sum_{S:\ d(S,T)\leq D} F(S),$$

where T is the target structure, i.e., an opened or closed hairpin, and $F(S)$ is the frequency of structure S. This frequency should be maximized to obtain the best sequence.

In calculating the frequencies for selectivity and ordinality, the search looks for sub-optimal structures using the algorithm of Wuchty et al. [18]. A structure is sub-optimal if its energy is lower than

$$[\text{minimum free energy}] + h.$$

By adopting only sub-optimal structures, we can neglect structures present at low frequency.

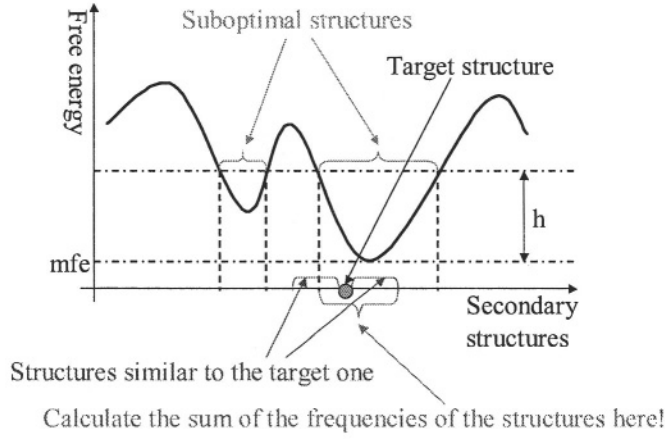


Fig. 6. The energy landscape of the DNA secondary structures of a sequence

3 Experiment

Based on the criteria introduced in the previous section, a structure design program has been implemented. This section explains the software and the result of its execution.

3.1 Programming

We have developed a secondary structure design program called `DNAhairpin`, in the C programming language, which uses the library of the Vienna RNA Package [7] mainly for thermodynamic calculations.

The original library of the Vienna RNA Package does not support the hybridization of multiple DNA strands. `DNAhairpin` concatenates multiple strands with virtual bases and regards them as a single strand using functions in the library. We modified several library functions, so that the effect of a loop structure built from virtual bases is ignored in the thermodynamics calculation.

Thermodynamic Parameters The thermodynamic parameters included in the Vienna RNA Package are for RNA molecules only. The parameters for DNA molecules were obtained by referring to reported physicochemical analyses of DNA hybridization.

The thermodynamic parameters required in the library are listed below.

- The free energies and enthalpies of stacked pairs [13].
- The free energy of the interaction between the closing pair of an interior loop and the two unpaired bases adjacent to the helix [1, 2, 3, 4, 12].

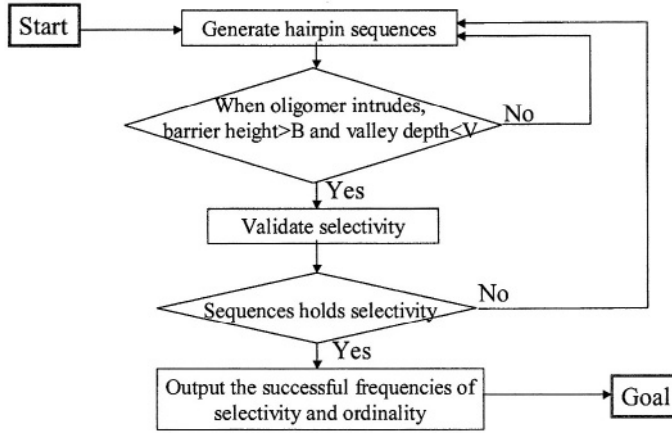


Fig. 7. The flowchart of the procedure used to design the secondary structures in a DNA hairpin

- The free energy of the interaction between the closing pair of a hairpin loop and the two unpaired bases adjacent to the helix.
- The enthalpies corresponding to these two cases [1, 2, 3, 4, 12].
- The free energies and enthalpies of the interaction of an unpaired base on the 5'-side that is adjacent to a helix forming multiple loops and the free ends [5].
- The free energies and enthalpies of the interaction between an unpaired base on the 3'-side that is adjacent to a helix forming multiple loops and the free ends [5].
- The free energies and enthalpies of symmetric interior loops of size 2.
- The free energies and enthalpies of interior loops of size 3 (2+1).
- The free energies and enthalpies of symmetric interior loops of size 4.
- The free energies of hairpin loops as a function of their size.
- The free energies of bulge loops as a function of their size.
- The free energies of internal loops as a function of their size.

For the items followed by citations, the program uses the parameters in the corresponding papers. The remaining parameters are estimated from other available parameters or use the corresponding thermodynamic parameters for RNA.

Details of the Procedure The procedure is essentially a "trial and error" method. Nevertheless, the order in which the criteria are checked requires some thought. The criterion that is the easier to check should be in the earlier step in the procedure. Following this policy, the first criterion to be checked is the barrier height and valley depth to check the ordinality, and the next criterion is the selectivity.

Following the selectivity check, the frequencies of structures that satisfy the selectivity and ordinality are calculated. This criterion is not used to screen sequences. The frequencies of the found sequences are printed out, and then the program stops regardless of the values. The program operator then needs to screen these based on the frequencies of the structures, since it takes time to calculate the frequencies and it is difficult to determine the exact thresholds of the frequencies.

The program runs efficiently on a parallel computer using multi-processing. The parallel version of the program is based on a simple "fork and join model" using independent processes at a coarse granularity.

3.2 Resulting Sequences

We present one of the best sets of sequences with the raw output generated by the program.

```
ordinality min barrier height: 4.369999
ordinality max valley depth: 2.650002

((((((((((((((((((((.......))))))))))))))))))))
CGAATACTCCTAACGTTGCGCGTTTGACGCAACGTTAGGAGTATTCG -23.440001
CTTTGAACACGTGAGTGGCATTACGAGTGCCACTCACGTGTTCAAAG -23.879999
CTTGTCCACCACGTTATGTTTCCGGGTAACATAACGTGGTGGACAAG -22.090000

selectivity (open): 0.767506 0.868699
selectivity (closed): 0.486210 0.733504
ordinality: 0.000384 0.022198
```

The threshold energy, *B,* of the barrier height is 4.3 kcal/mol and the threshold energy, *V,* of the valley depth is 2.9 kcal/mol. The size of the hairpin structure can be specified using the command line arguments. In this example, the default lengths of the head of the oligomer, hairpin stem, and hairpin loop are 10, 20 and 7, respectively. The threshold distance, *D,* of the structural similarity equals the length of the hairpin stem. The threshold energy, h, of the structural suboptimality is 3.00 kcal/mol for the selectivity check, and 6.00 kcal/mol for the ordinality check in order to raise the precision of the minimum frequency value for the ordinality.

First, the minimum barrier height and maximum valley depth of transition paths violating the ordinality are printed out. Then, the set of sequences that satisfies the selectivity and the criteria for the transition paths for ordinality are output. The sequences are followed by the free energies (kcal/mol) of the hairpin structure.

Finally, the frequencies of the target structures are listed, including suboptimal ones. The value `0.767506` is the minimum frequency of the target structure in Figure 3 (1-a) for all combinations of the three sequences. Conversely, `0.868699` is the maximum value. In the last two lines, the target structure is the

closed structure (Figure 3 (2-b), Figure 4 (A) and (B)). The penultimate line gives the selectivity, and the last line the ordinality.

4 Discussion and Conclusion

The ultimate goal of this study is to implement DNA nanomachines that are more sophisticated than existing ones. First, our project team proposed a multi-state molecular machine that sequentially changes the conformation of repeated DNA hairpin structures. This paper suggests a novel method for designing such multi-state molecular machines.

Existing methods are insufficient to construct such a huge molecular machine composed of multiple DNA molecules. The method proposed here verifies the changes in their formation exhaustively, but efficiently, by dividing the secondary structure into hairpin structure units. It considers the minimum free energy structure, the structure transition path, and the total frequency of target structures, including sub-optimal ones. Hence, it can design base sequences that are more appropriate thermodynamically. We have implemented the secondary structure design method as a C program.

Our system considers the free energies of structures primarily. Structure transition is also dealt with as a series of free energies. A model using only free energies is rather simple and requires a moderate amount of computing power compared to a more accurate model based on kinetics. However, it lacks much information about molecular dynamics and cannot calculate the "true energy barrier height". To design a more robust secondary structure, it is necessary to combine our method of structure design with detailed kinetic analyses, such as molecular dynamics.

It is also necessary to examine whether the criteria that we use to evaluate the sequences are valid. A laboratory experiment is one of the most practical ways to examine the criteria. Using sequences that we designed, our project team [8] has performed several laboratory experiments. More experiments are needed to validate the method used to design the secondary structure. Moreover, the results of these laboratory experiments should be used as feedback to further improve the design method.

We also plan to incorporate other aspects of the structure of DNA into the model. Various physical properties of double stranded DNA inhibit hybridization [14, 15, 16, 19]. To consider some of these properties, we must go beyond an examination of secondary structure.

Acknowledgements

Among others, we thank Kensaku Sakamoto, Masahito Yamamoto, and Atsushi Kameda for designing and implementing the hairpin machine. We also thank an anonymous reviewer for a valuable comment that improved this paper. This work was supported by JST CREST and by the Ministry of Education, Culture, Sports, Science, and Technology of Japan under Grants-in-Aid for Scientific Research on Priority Areas (B) 14085101 and 14085202, 2003.

References

[1] Hatim T. Allawi *et al.*: Thermodynamics and NMR of internal GT mismatches in DNA, *Biochemistry* 36, 10581–10594, 1997.

[2] Hatim T. Allawi *et al.*: Nearest neighbor thermodynamic parameters for internal GA mismatches in DNA, *Biochemistry* 37, 2170–2179, 1998.

[3] Hatim T. Allawi *et al.*: Thermodynamics of internal CT mismatches in DNA, *Nucleic Acids Research* 26, 2694–2701, 1998.

[4] Hatim T. Allawi *et al.*: Nearest-neighbor thermodynamics of internal AC mismatches in DNA: Sequence dependence and pH effects, *Biochemistry* 37, 9435–9444, 1998.

[5] Salvatore Bommarito *et al.*: Thermodynamic parameters for DNA sequences with dangling ends, *Nucleic Acids Research* 28, 1929–1934, 2000.

[6] Christoph Flamm *et al.*: RNA folding at elementary step resolution, *RNA* 6, 325–338, 2000.

[7] Ivo L. Hofacker *et al.*: Fast folding and comparison of RNA secondary structures, *Monatshefte für Chemie (Chemical Monthly)* 125, 167–188, 1994.

[8] Atsushi Kameda *et al.*: Conformational addressing using the hairpin structure of single-stranded DNA, *in this volume.*

[9] Chengde Mao *et al.*: A nanomechanical device based on the B-Z transition of DNA, *Nature,* 397, 144–146, 1999.

[10] J. S. McCaskill: The equilibrium partition function and base pair binding probabilities for RNA secondary structure, *Biopolymers,* 29, 1105–1119, 1990.

[11] Steve R Morgan *et al.*: Barrier heights between ground states in a model of RNA secondary structure, *J. Phys. A: Math. Gen.* 31, 3153–3170, 1998.

[12] Nicolas Peyret *et al.*: Nearest neighbor thermodynamics of DNA with AA, CC, GG and TT mismatches, *Biochemistry* 38, 3468–3477, 1999.

[13] John SantaLucia, Jr.: A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics, *Proc. Natl. Acad. Sci. USA,* 95, 1460–1465, 1998.

[14] Friedrich C. Simmel *et al.*: Using DNA to construct and power a nanoactuator, *Physical Review E,* 63, 041913, 2001.

[15] Friedrich C. Simmel *et al.*: A DNA-based molecular device switchable between three distinct mechanical states, *Applied Physics Letters,* 80, 883–885, 2002.

[16] A. J. Turberfield *et al.*: DNA fuel for free-running nanomachines, *Physical Review Letters,* 90, 11, 118102, 2003.

[17] Hiroki Uejima *et al.*: Analyzing the secondary structure transition paths of DNA/RNA molecules, *in this volume.*

[18] Stefan Wuchty *et al.*: Complete suboptimal folding of RNA and the stability of secondary structures, *Biopolymers,* 49, 145–165, 1999.

[19] Hao Yan *et al.*: A robust DNA mechanical device controlled by hybridization topology, *Nature,* 145, 62–65, 2002

[20] Bernard Yurke *et al.*: A DNA-fuelled molecular machine made of DNA, *Nature,* 406, 605–608, 2000.

[21] M. Zuker *et al.*: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, *Nucleic Acids Research* 9, 133–148, 1981.

Analyzing Secondary Structure Transition Paths of DNA/RNA Molecules

Hiroki Uejima and Masami Hagiya

Japan Science and Technology Corporation (JST-CREST) and
Department of Computer Science,
Graduate School of Information Science and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
{uejima, hagiya}@is.s.u-tokyo.ac.jp

Abstract. Analysis of secondary structure transition is important for designing bistable DNA/RNA molecules. To make reliable molecular machines out of such molecules, it is necessary to estimate the height of the energy barrier to the structure transition. This paper suggests three kinds of optimized transition path: the locally optimized direct path, the globally optimized direct path, and the globally optimized path. These paths can be used as criteria for evaluating structure transitions. Then, we introduce algorithms to obtain or approximate these optimized paths. The algorithm that Morgan and Higgs used to obtain the globally optimized direct path is analyzed and improved as an algorithm on a bipartite graph. The algorithms are implemented as a C language program.

1 Introduction

The height of the energy barrier on a transition path between two secondary structures reflects the efficiency of the transition between the structures, and can be used in designing a molecular machine with multiple stable structures. We used the energy barrier height for secondary structures to design a molecular machine made of DNA [4]. Since the energy barrier height depends on the structure transition path, it is necessary to define the characteristic transition path in order to determine a meaningful value for the energy barrier height.

There are two tractable ways to approximate the energy barrier height. One method generates transition paths randomly and finds their lowest energy mountaintop. A transition path is generated, as transitions proceed more frequently in the direction in which the increase in the free energy is smaller. This probabilistic algorithm was proposed by Flamm *et al.* [1]

The other method takes advantage of the heuristics proposed by Morgan and Higgs [3] based on a simplified energy model of secondary structures. This model uses the number of base pairs as the free energy of a structure. Their heuristics can generate a path with a very low energy mountaintop efficiently, but this method guarantees nothing about the properties of the path. Therefore, it is necessary to generate a number of paths using this heuristic to obtain a more accurate approximation of the exact barrier height.

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 86–90, 2004.

2 Formalism

For set X of base sequences of DNA strands, $S(X)$ denotes the set of all secondary structures formed by the sequences in X, where a secondary structure is represented as a set of base pairs from X. For a given structure s in $S(X)$, the free energy function E gives the free energy $E(s)$ of s based on a thermodynamic model, such as the nearest neighbor model.

The *structure transition path* p from an initial structure s_0 to a final structure s_n is a series of structures $p = <s_0, \ldots, s_n>$, where $s_i \in S(X)$ and $d(s_i, s_{i+1}) = 1$. $d(s_i, s_{i+1})$ denotes the distance between structures s_i and s_{i+1}, i.e., it gives the size of the symmetric difference between s_i and s_{i+1} as sets of base pairs. The length $|p|$ of the structure transition path p is $|p| = n$, and the i-th structure s_i of p is denoted by $p[i]$.

If the length $|p|$ of path p from structure s to structure s' is $|p| = d(s, s')$, then the path p is called *direct.*

The *energy mountaintop height* $E_{\text{top}}(p)$ of a path p is

$$E_{\text{top}}(p) = \max_i (E(p[i]) - E(p[0])).$$

A locally optimized direct path is a direct path that is built in such a greedy way that every energy increment along the path is minimized locally. A direct path p of length n is called a *locally optimized direct path* (LDP), if for any direct path q from $p[0]$ to $p[n]$ and for any i $(1 \leq i \leq n)$,

$$q[i-1] = p[i-1] \Rightarrow E(p[i]) - E(p[i-1]) \leq E(q[i]) - E(q[i-1]).$$

A direct path is called a *globally optimized direct path* (GDP), if its energy mountaintop height is the smallest of all direct paths.

A *globally optimized path* (GP) is the path whose mountaintop height is the smallest of all of the paths.

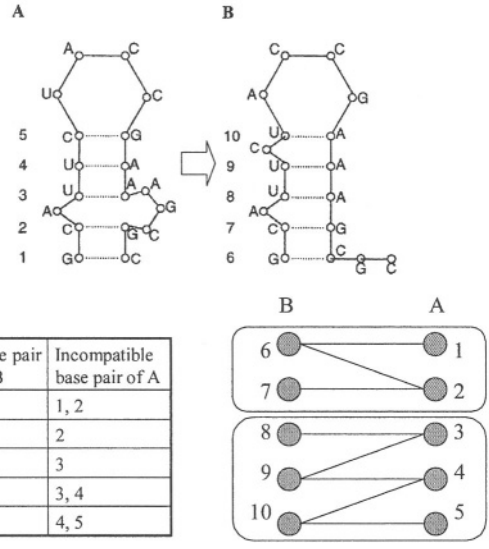
3 Morgan and Higgs' Heuristic and Improvement

The heuristic proposed by Morgan and Higgs [3] is based on a simple model in which the number of base pairs in a structure determines its free energy. In short, the free energy of structure s is approximated as:

$$E(s) = -[\text{the number of base pairs of } s].$$

Their algorithm tries to find the direct path along which structures maintain as many base pairs as possible during the transition. It finds a relatively good solution, but it never guarantees the optimality of the solution.

The concept of "incompatible" base pairs in the initial structure is introduced before explaining the algorithm itself. If base pair ω in the initial structure is required to be broken in order to form base pair $\tilde{\omega}$ in the final structure, ω is incompatible with base pair $\tilde{\omega}$. The incompatibility of base pairs is caused by either sharing a base or avoiding a pseudoknot structure. Using this notion of incompatible base pairs, the algorithm is described in the following way.



Base pair of B	Incompatible base pair of A
6	1, 2
7	2
8	3
9	3, 4
10	4, 5

Fig. 1. An example of two transition structures and their incompatibility relationship [3]. The incompatible bases involved in changing structure A into structure B are listed in the table. The incompatibility relationship can be represented by a bipartite graph in this figure

1. Ω = [the set of base pairs in the initial structure],
   $\tilde{\Omega}$ = [the set of base pairs in the final structure].
2. Choose the base pair $\omega \in \tilde{\Omega} \setminus \Omega$ that has the fewest incompatible base pairs with respect to the current values of Ω and $\tilde{\Omega}$. (If there is more than one such base pair, choose one randomly.)
3. $\Omega := (\Omega \setminus \{\omega'_1, \ldots, \omega'_m\}) \cup \{\omega\}$, where $\omega'_1, \ldots, \omega'_m$ are the base pairs incompatible with ω. In other words, base pairs $\omega'_1, \ldots, \omega'_m$ are removed, and base pair ω is added to the current structure.
4. If $\Omega = \tilde{\Omega}$ then stop. Otherwise, return to Step 2.

This algorithm generates a candidate GDP. In practice, the algorithm is applied to one problem several times, and the best solution, the one with the lowest mountaintop, is picked as an approximate solution of the GDP.

The incompatibility relationship of base pairs can be represented using a bipartite graph called a *base pair incompatibility graph* (Figure 1). Its vertices correspond to base pairs and each of its edges represents an instance of the incompatibility relation. If a base pair is incompatible with another base pair, there is an edge between the corresponding vertices in the base pair incompatibility graph. Since there are no edges between the vertices of base pairs belonging to the same structure, the graph is bipartite.

Morgan and Higgs' algorithm (M-H algorithm) is translated into a procedure on a base pair incompatibility graph in the following manner. Let U and V be the

sets of vertices corresponding to the initial structure *A* and the final structure *B*, respectively.

1. Choose the vertex $v \in V$ whose degree is the smallest of all the vertices in *V*.
2. Remove the vertices connected to v, and remove the edges incident to the vertices.
3. Remove all the vertices in *V* whose degree is zero.
4. If all the vertices have been removed, then stop. Otherwise, return to Step 1.

Note that the M-H algorithm does not specify the order of removing the vertices in *U*.

It is possible to improve the algorithm by applying the procedure to each connected component of the bipartite graph. Since all of the connected components are independent, the order of processing the components does not affect the processing of each component. If the minimum mountaintop height corresponding to each component has been obtained, the optimal order for obtaining the lowest mountaintop for the entire transition is as described below.

1. First, process the components such that $|U_i| - |V_i| < 0$ in ascending order of their mountaintop height. If the mountaintop heights of some components are equal, process them in the descending order $||U_i| - |V_i||$.
2. Then, process the components such that $|U_j| - |V_j| = 0$.
3. Finally, process the components such that $|U_k| - |V_k| > 0$ in descending order of their mountaintop height. If the mountaintop heights of some components are equal, process them in the ascending order $|U_k| - |V_k|$.

$|U_i|$ denotes the size of the set U_i. $|U_i| - |V_i|$ is the energy height of the final structure above the initial level.

The path obtained by the M-H algorithm is not necessarily optimal, as shown by the example (Figure 2). In the bipartite graph in Figure 2, a blue edge denotes the incompatibility relation caused by sharing a base, while a red edge denotes the incompatibility relation caused by avoiding a pseudoknot structure.

4 DNAtrans

We implemented a local search for the LDP and the M-H algorithm for the GDP as a C language program called DNAtrans using the Vienna RNA Package [2]. It calculates the locally optimized direct path (LDP) and approximates the globally optimized direct path (GDP) of the structure transition between two given structures of a single sequence. The approximation of the GDP is based on the M-H algorithm.

The GDP approximated using the M-H algorithm can be compared with the LDP by calculating the free energy of each structure along the GDP using the nearest neighbor thermodynamic model. According to an experimental result, the energy mountaintop height of the LDP obtained using a local search is close to and often a bit smaller than the GDP approximated using the M-H algorithm.

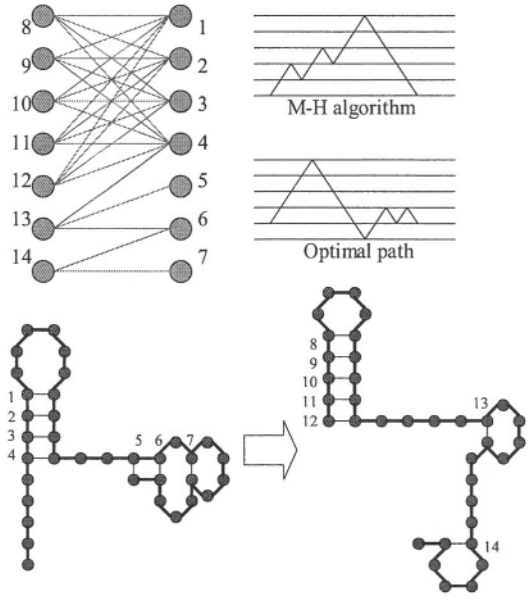


Fig. 2. An example in which the path obtained by the M-H algorithm is not optimal, its base pair incompatibility graph, and the transition of its energy height

Judging from this result, the energy landscape around such a simple structural transformation seems relatively smooth.

The execution times of the algorithms used to obtain the LDP and GDP are both less than one second if the structure distance is less than one hundred. Although repeated application of the M-H algorithm to increase the accuracy of the GDP solution may take time, it is in general better to adopt the GDP to obtain lower barrier heights.

References

[1] Christoph Flamm *et al.*: RNA folding at elementary step resolution, *RNA* 6, pp.325–338, 2000.

[2] Ivo L. Hofacker *et al.:* Fast folding and comparison of RNA secondary structures, *Monatshefte für Chemie (Chemical Monthly)* 125, pp.167–188, 1994.

[3] Steve R Morgan *et al.:* Barrier heights between ground states in a model of RNA secondary structure, *J. Phys. A: Math. Gen.* 31, pp.3153–3170, 1998.

[4] Hiroki Uejima *et al.:* Secondary Structure Design of Multi-state DNA Machine Based on Sequential Structure Transitions, *in this volume.*

Self-Assembled Circuit Patterns

Matthew Cook, Paul W.K. Rothemund, and Erik Winfree

Computer Science and Computation & Neural Systems
California Institute of Technology, Pasadena, CA 91125, USA

Abstract. Self-assembly is a process in which basic units aggregate under attractive forces to form larger compound structures. Recent theoretical work has shown that pseudo-crystalline self-assembly can be *algorithmic,* in the sense that complex logic can be programmed into the growth process [26]. This theoretical work builds on the theory of two-dimensional tilings [8], using rigid square tiles called Wang tiles [24] for the basic units of self-assembly, and leads to Turing-universal models such as the Tile Assembly Model [28]. Using the Tile Assembly Model, we show how algorithmic self-assembly can be exploited for fabrication tasks such as constructing the patterns that define certain digital circuits, including demultiplexers, RAM arrays, pseudowavelet transforms, and Hadamard transforms. Since DNA self-assembly appears to be promising for implementing the arbitrary Wang tiles [30, 13] needed for programming in the Tile Assembly Model, algorithmic self-assembly methods such as those presented in this paper may eventually become a viable method of arranging molecular electronic components [18], such as carbon nanotubes [10, 1], into molecular-scale circuits.

1 Introduction

A simple example of embedding computation in self-assembly is shown in Figure 1 (from [29]). The seven square tiles pictured in Figure 1(a) are Wang tiles [24]; they are to be arranged so that labels on the sides of abutting tiles match. Many copies of each tiles may be used, but the tiles may not be flipped or rotated. The result is a pattern such as the one shown in Figure 1(c).

To be applicable to the subject of self-assembly, Wang's tiling model must be extended to describe *how* the tiles aggregate into patterns, based on simple local rules. The Tile Assembly Model [28] does this by assigning an integer *bond strength* to each side of each tile. Growth occurs by the addition of single tiles, one at a time. In order for a new tile to attach itself to an existing pattern of tiles, the sum of the bond strengths on the edges where it would stick must sum to at least the threshold τ, a fixed parameter of the experiment.

The tiles shown in Figure 1(a) constitute a *self-assembly program* for counting in binary, and we will refer to them in this paper as the *counter tiles.* Lines on the edges are drawn to indicate the strength of binding: a thin line indicates a strength-1 bond, thin double lines indicate a strength-2 bond, and a thick line indicates a strength-0 bond (i.e. a side that does not stick to anything). Of course, a bond is formed only when the edge labels match.

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 91–107, 2004.

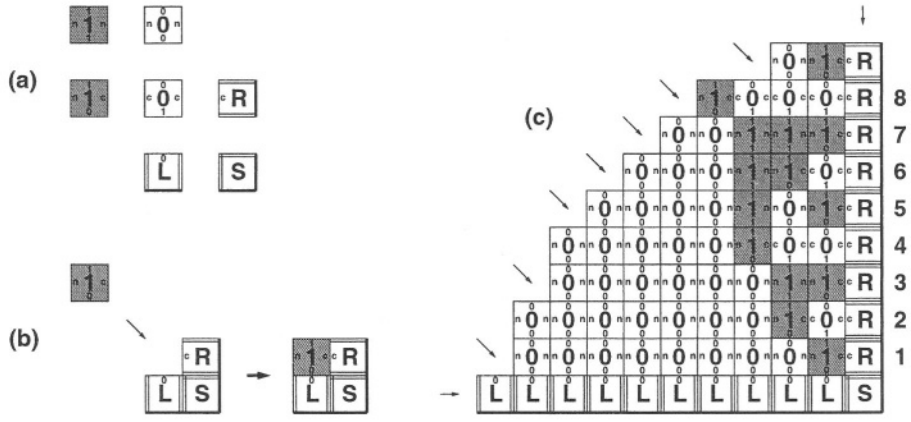


Fig. 1. The counter tiles (from [29]). The set of seven tiles shown in (a) are a Tile Assembly Model program for counting in binary. The tiles labeled "1" are colored gray to make it easier to see the resulting pattern, visible in (c). The self-assembly progresses by individual tiles accreting to the assembly as shown in (b). Edges marked with a small letter or number have bond strengths of 1, while edges with a double line have bond strengths of 2 (and do not require a further label here, since there is only one vertical and one horizontal kind). A later stage of self-assembly is shown in (c), with arrows indicating all the places that a new tile could accrete

To understand how the program works, we can conceptually categorize the seven tiles used in this example into two groups: The three tiles bearing large letters, called *boundary tiles,* are used to set up the initial conditions on the boundary of the computation. The four tiles bearing large numbers, called *rule tiles,* perform the computation and their numbers are to be interpreted as the binary digits of the output pattern.

The pattern in Figure 1(c) shows a stage of self-assembly with $\tau = 2$, so tiles can only bind to one another when the total binding strength is ≥ 2. For example, an "L" tile may bond on either side to another "L" tile or on its right side to an "S" tile, using a single strength-2 bond. The rule tiles, which can form only strength-1 bonds, can only bind to an assembly if two or more bonds cooperate to hold the tile in place, since $\tau = 2$. Thus, at first, the only counter tiles which can assemble are boundary tiles, via strength-2 bonds. Only after the boundary tiles have begun to assemble into a V-shape, can rule tiles begin binding at *corner sites* as shown in Figure 1(b). The rule tile shown there can form two strength-1 bonds, and it is the only tile that can stick there.

Successive additions of rule tiles and boundary tiles would result in a structure like that in Figure 1(c) whose rows may be read, from bottom to top, as an enumeration of binary numbers. To understand how this works, inspect the rule tiles. Consider the bottom and right sides of each rule tile as *inputs,* and the left and top sides as *outputs.* A rule tile fitting into a corner "reads" two input bits by matching bonds; one bit it reads is the identity of the digit below it and the other is the carry bit from the tile to its right (if "c", carry= 1; if "n",

carry= 0). The number on the rule tile and the bond that it outputs on its top reflect the result of adding, modulo 2, the two input bits; the bond it outputs to its left reflects the resulting carry bit. Rule tiles thus copy the digits below them, unless a carry is indicated from the right. Initially the "L" boundary tiles present all zeros to the rule tiles from below; this starts the counting at zero. The "R" boundary tiles present a new carry bit for each row of the counter from the right; this adds 1 to each successive row of the counter.

It is clear from Figure 1(c) that multiple corner sites may be available for binding rule tiles at the same time; the order in which tiles are added at these sites is not specified. Despite the nondeterministic nature of assembly, it can be shown that the infinite structure that is formed by the counter tiles is unique [27]. This is essentially because a unique rule tile binds at each corner site. For the counter tiles, this in turn is a consequence of our requirement that a rule tile may be added only by the cooperative formation of at least two bonds at once, that is, $\tau = 2$ while the rule tile bond strengths are each 1.

To understand why we use $\tau = 2$, consider what would happen if $\tau = 1$. Since bond strengths are required to be integers, any Tile Assembly Model program with $\tau = 1$ would not be able to require a tile to match the assembly on more than a single side, which makes information processing difficult at best, and if a unique output is required, self-assembly at $\tau = 1$ appears not to be Turing-universal in two dimensions.

If $\tau = 2$ is more powerful than $\tau = 1$, then why don't we try even higher values? The two-fold answer is that (A) there does not seem to be much to gain, since most Tile Assembly Model programs already work well with $\tau = 2$, and (B) the experimental conditions must allow a tile to be able to distinguish between a total bond strength of τ vs. a total bond strength of $\tau - 1$, so experimentally it is good to maximize the ratio between these, which means minimizing τ.

Is the $\tau = 2$ assumption reasonable for physical systems? Real crystal growth does seem to approximate $\tau = 2$ growth with strength-1 bonds. The phenomena of faceting and supersaturation are both consequences of the rarity of steps that violate $\tau = 2$. If a *programmable* experimental system well-modeled by $\tau = 2$ (such as [30] or [19]) can be perfected, then two-dimensional self-assembly can be used to build a binary counter, and in fact, two-dimensional $\tau = 2$ self-assembly is universal [26]. That is, any computer program may be translated into a set of tiles that when self-assembled, simulate the computer program. But the stubbornly practical may still ask: *What is such an embedding of computation in self-assembly good for?*

2 Self-Assembled Circuits

In principle we could use self-assembly wherever we use a conventional computer. In practice we do not expect that computation by self-assembly will be able to compete with the speed of traditional computer architectures on explicitly computational problems. Instead, factors such as the physical nature of the output and the ability to run the same program many times at once in parallel motivate

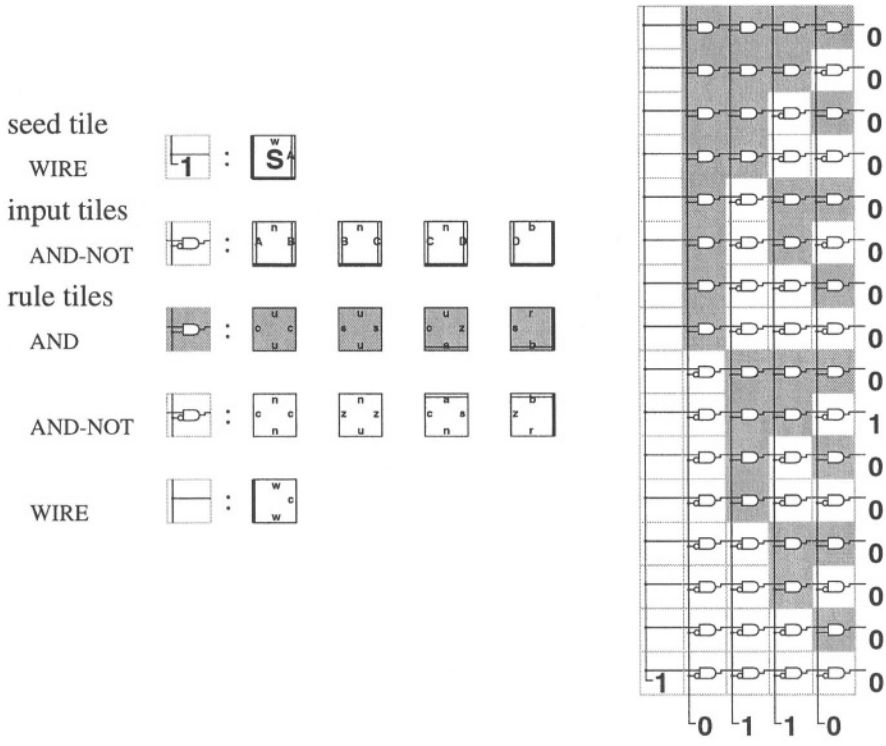


Fig. 2. Using a binary counter to self-assemble a demultiplexer. Logic levels for an example input-output pair are shown: only the row that exactly matches the input pattern is set to "1". To make a pattern with N rows, 10 + log N tiles are used

us to look for *fabrication problems:* particular patterns or sets of patterns that have potentially useful properties (e.g. as templates for electronic circuits), and which are amenable to self-assembly.

Naively we might wonder, "Can we self-assemble the circuit for a contemporary CPU?" Assuming that we can create tiles that act as circuit elements[1] what we are really asking is "Can we self-assemble the layout pattern for a CPU?" The answer, in theory, is yes, and we may do so without using any complex computation.

Any particular pattern, no matter how complex, can be self-assembled by assigning a unique tile type, with a unique set of binding interactions with its neighbors, to each position in the pattern. The resulting program is as big as the pattern itself, with every tile in the program being used just once in the pattern. This type of self-assembly program (called *unique addressing*) is undesirable because it is not efficient — an efficient program would use a small number of tile

[1] Periodic electrical networks of functional LEDs have already been self-assembled on the millimeter scale [7].

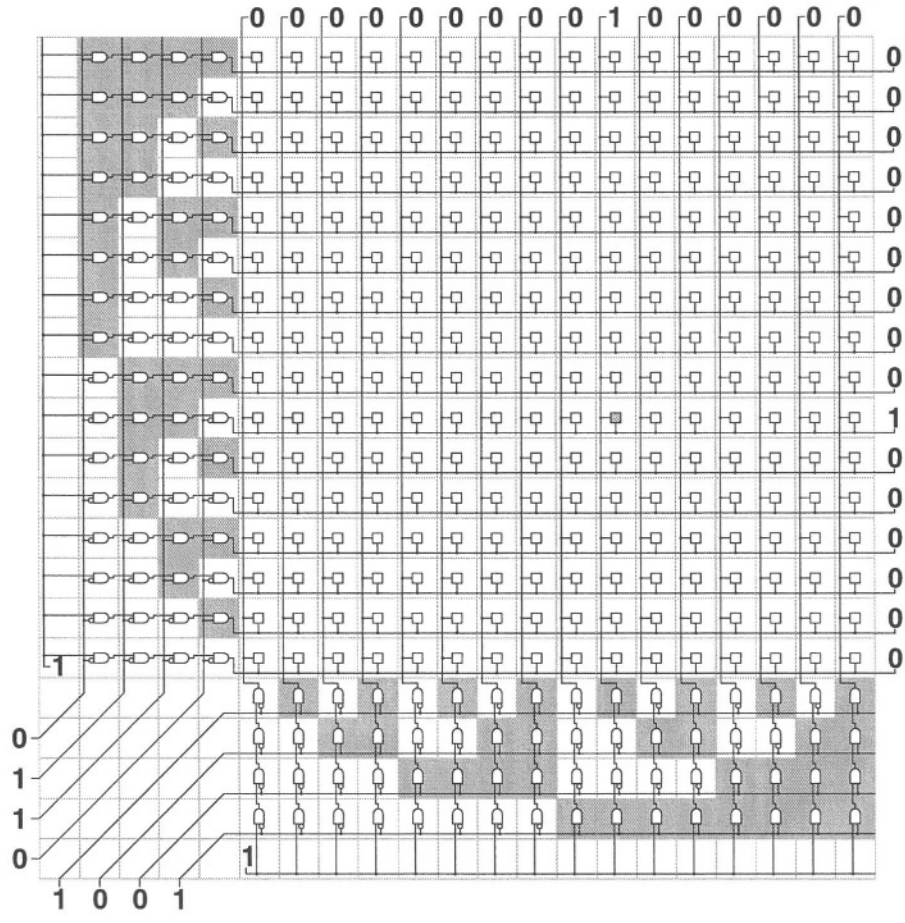


Fig. 3. Two self-assembled demultiplexers at right angles can address a memory. The gray memory cell is being addressed in this figure

types compared to the size of the pattern. Instead, unique addressing uses the greatest number of tile types possible to create a pattern. In physical implementations [30] it appears that creating unique tile types and unique specific binding interactions is expensive and difficult, so with currently-envisioned techniques it seems that unique addressing is impractical except for very small patterns.

For a circuit to be well-suited to self-assembly, its structure should have a highly methodical pattern to it. The simplest such pattern would be a periodic arrangement of units, such as occurs in a random-access memory circuit, shown in the upper right region of Figure 3. Indeed, using DNA self-assembly to create a molecular-scale memory was suggested in [18]. The pattern generated by the counter tiles of Section 1 is a somewhat more interesting pattern, yet still

methodical, which we can see is why it was easy to implement via self-assembly. Later in this paper we will encounter more circuits with methodical structure.

Looking again at the counter tiles, we can think about what similar programs we might be able to construct. The counter tiles use a constant number of tile types to form a structure that grows indefinitely in two directions. If we wish to form a structure of a specific chosen size, we need a set of tiles that not only count, but also stop when the count is complete. Such efficient self-assembly programs for growing finite shapes have been presented in [20]. Here, we use an improved construction [5] wherein, in each successive row, the rightmost "0" is replaced by a "1" and all bits to its right are zeroed. If there is no rightmost "0", it stops. In this construction, shown in Figure 2, a set of $\log N$ *input tiles* are used to define the width of the counter; the assembly grows into a rectangle of exactly size $N \times (1 + \log N)$. Thus the counter can be used to make relatively narrow structures of a chosen length. By adding an additional constant number of tiles we can self-assemble $N \times N$ squares. In these examples, wedding computation with self-assembly addresses what is to chemists a difficult synthetic (fabrication) problem — how to make polymer or crystalline structures of a well-defined size.

Perhaps surprisingly, the binary counter itself happens to yield the layout for a useful circuit. In Figure 2, each tile type is shown labelled with a circuit element, such as a wiring arrangement, an AND gate, or an AND-NOT gate. Once assembled, the tiles form a circuit with 4 input lines along the bottom and $2^4 = 16$ output lines along the right. This is a demultiplexer: the address bits on the input lines specify exactly one output line to be active. A larger circuit with $n = \log N$ input lines and $N = 2^n$ output lines can be self-assembled by changing only the n input tiles. Note that multiple types of tiles can carry the same circuit element. This is a common phenomenon: all the markings on all the tiles comprise rather more information than just the pattern that we care to create; this excess information is necessary to specify how to grow the pattern correctly.

This is our first example of self-assembly being used to create a useful circuit[2]. Whether or not this could be practical depends upon how the tiles are implemented physically and how the circuit elements are attached to the tiles. Let's speculate on a few possible approaches, each of which involves considerable challenges. For example, if the tiles were made of DNA (e.g., the 2×12 nm molecules in [30]) and the circuit elements were small molecular electronic devices (e.g., [6, 14]) covalently attached to the DNA, some chemical post-processing could be necessary to make functional connections between the circuit elements. On the other hand, if again DNA tiles were used but now the labels were single-stranded DNA protruding from the tiles, then in a post-processing step after assembly is complete, larger circuit elements (e.g., DNA-labelled carbon nanotubes [25]) could be arranged by hybridization to the self-assembled pattern,

[2] Our approach, in which the self-assembled patterns are used as templates for fabricating functional circuits out of other materials, can be contrasted to work that uses the self-assembly process itself to perform either a fixed [12] or reconfigurable [4] computation.

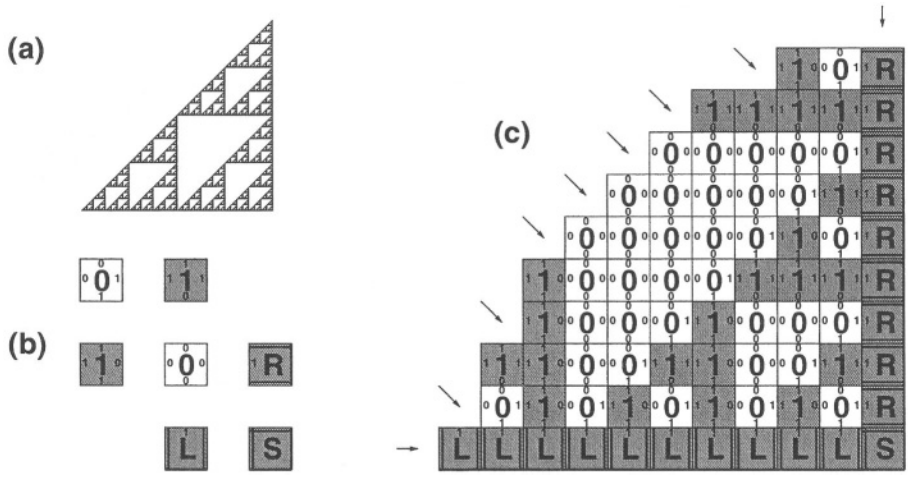


Fig. 4. The Sierpiński triangle and a set of tiles that construct it in the limit

thereby forming the desired circuit. Alternatively, the tiles could be micron- or millimeter- scale objects with embedded conventional electronic components, as in [7, 11, 3]. Algorithmic self-assembly has been demonstrated at this scale as well [19].

A demultiplexer could be used as a building block for a larger self-assembled circuit: a pair of demultiplexers oriented at right angles along the borders of an $N \times N$ memory allow a memory element to be accessed using only $2 \log N$ lines. Thus a memory circuit may be self-assembled (see Figure 3). What other circuits might be possible? Our next constructions derive from the observation that the demultiplexer circuit implements a generalized inner product of a binary vector by a binary matrix, with the binary function EQUALS substituting for multiplication and AND substituting for addition in the definition of matrix multiplication. That is, the circuit takes an n-bit binary vector, "multiplies" it by a $n \times 2^n$ size binary "counting" matrix, and outputs a 2^n long vector. Similarly, a circuit for an arbitrary binary matrix multiplication could be created by self-assembling a circuit decorated with logic gates as appropriate for the matrix of choice.

3 Self-Similar Transforms

Another complex pattern that may be created by a simple self-assembling computation is the Sierpiński triangle, pictured in Figure 4(a). Only seven tiles, shown in Figure 4(b) (from [27]), are required to create a pattern (shown in Figure 4(c)) whose limit is this triangular fractal pattern. As with the counter tiles, its construction depends on $\tau = 2$ assembly. By labeling the sides of the tiles as "input" and "output", individual tiles can be seen to encode the binary function XOR. Diagonals of the assembly, interpreted as zeros and ones, form rows of

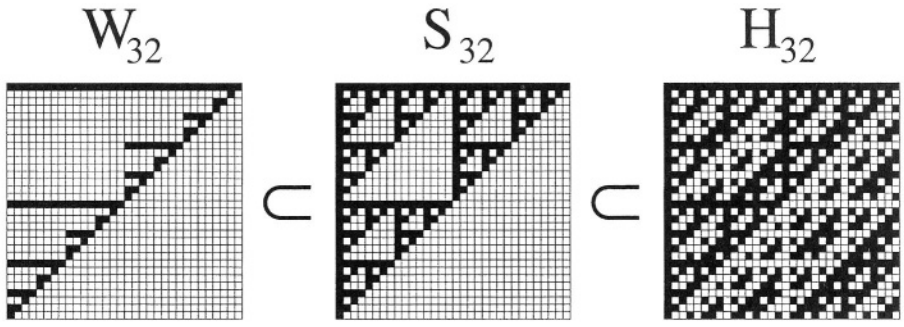


Fig. 5. Comparison of self-similar binary matrices. Prom left to right, the binary pseudowavelet matrix (W_{32}), the Sierpiński triangle matrix (S_{32}), and the Hadamard matrix (H_{32}). For the pseudowavelet and Sierpiński matrices, black represents 1 and white represents 0. For the Hadamard matrix, black represents 1 and white represents -1

Pascal's triangle modulo 2. It can also be seen that diagonals of the assembly are instantaneous descriptions of a one-dimensional cellular automaton. Aside from its interpretation as a computation, this pattern is beginning to find some practical uses; rendered in metal the Sierpiński triangle appears to be a superior cellular phone antenna [17].

Does the Sierpiński triangle also have a circuit interpretation like the binary counter? Perhaps not, but it inspires thought: interpreted as a binary matrix the Sierpiński triangle has many periodic rows whose periods are related by a logarithmic scaling. This suggests that using the Sierpiński triangle as a matrix multiplier might effect some transform similar to a wavelet or Fourier transform. In fact, binary versions of the wavelet and Fourier transforms, namely the binary pseudowavelet transform [15] and the Hadamard transform [21, 16, 31], have self-similar matrices closely related to the Sierpiński triangle. Both these transforms have been used in signal processing and image compression. The Hadamard matrix in particular has uses from quantum computation to cell phones, and can be used directly for implementing a parallel Walsh transform [23]. Many theoretical and practical uses have been studied for Hadamard matrices of size 2^n [2,9,22].

Given the similarity of these transforms to the Sierpiński triangle, it seems reasonable to expect that there should exist simple tile sets that self-assemble into circuit patterns for computing them. This turns out to be correct.

In Figure 5 we give a visual comparison of these matrices to the Sierpiński matrix. Their formal similarity can be seen from their recursive definitions: $W_1 = T_1 = H_1 = S_1 = [1]$, and for n a power of 2,

$$W_{2n} = \begin{bmatrix} T_n & W_n \\ W_n & 0 \end{bmatrix}, \quad T_{2n} = \begin{bmatrix} T_n & T_n \\ 0 & 0 \end{bmatrix}, \quad S_{2n} = \begin{bmatrix} S_n & S_n \\ S_n & 0 \end{bmatrix}, \quad H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$$

The pseudowavelet transform W_n has a simple self-similar structure for which it seems likely we can find a simple self-assembly program; in fact, a straightforward modification of the Sierpiński tiles will suffice. First, modify the tiles

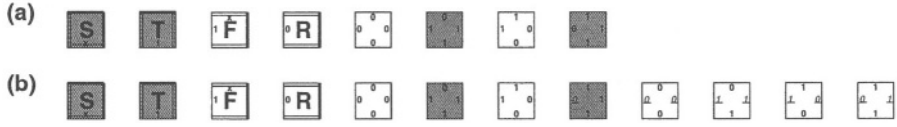


Fig. 6. Construction of the pseudowavelet tile set. (a) A tile set for growing the Sierpiński triangle from the upper right corner, as in Figure 5. (b) A tile set for growing the pseudowavelet transform from the upper right corner

Fig. 7. The two types of hexagonal tile that will be used for constructing the pattern in Figure 8(c)

so growth occurs from the right to the left (in Figure 5); then make a "tagged" version of each rule tile such that in each row, the first "0" to the left of a "1" gets tagged, and tags propogate leftward. The black cells are defined by only untagged tiles. These tiles are shown in Figure 6. Although these tiles build unbounded patterns, patterns of defined size can be created by replacing the 4 boundary tiles with a binary counter, as in Figure 2 and Figure 3.

4 Growing a Hadamard Matrix

In this section we will present a set of hexagonal tiles which deterministically constructs self-similar Hadamard matrices of order 2^n. We begin, in Subsection 4.1, by presenting a simple set of "red and green" tiles which constructs a nice but non-Hadamard self-similar pattern. Then, in Subsection 4.2 we will present a slight elaboration on those tiles which results in the generation of a Hadamard matrix and indicate how to turn the given construction into one that works with square tiles. Finally, in Subsection 4.3 we prove the correctness of our construction.

4.1 Red and Green Tiles

Figure 7 shows two hexagonal types of tiles, one red and one green. Unlike the square Wang tiles discussed in earlier sections, these tiles may be rotated and/or flipped over. Where two tiles abut, the notches on the sides of the hexagons must fit together (one out and one in), or, where there are spots instead of notches, the spots of the tiles must match. During growth, a new hexagon will need to

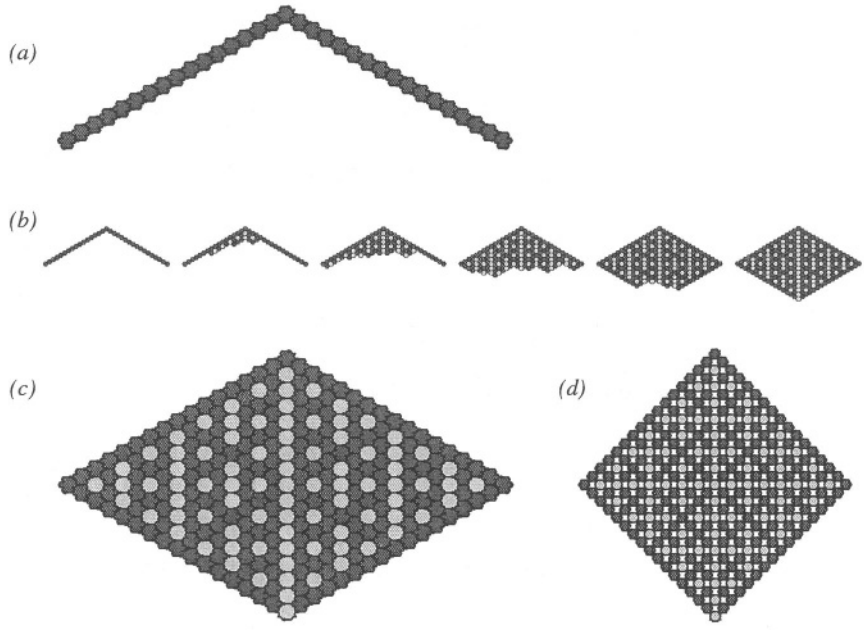


Fig. 8. Stages of growth for the tiles from Figure 7. (a) shows the boundary condition used to start the growth. (b) shows a sample sequence of snapshots as it grows over time. (c) shows the pattern after it has fully filled in. (d) shows exactly the same pattern as (c), but with the tiles pulled apart vertically so that the overall shape is now a square. The relationship to Figure 9 begins to become apparent

fit in with three existing hexagons, so we will have $\tau = 3$. (Later we will show how τ can be reduced to 2 by converting the hexagons into square tiles.)

The boundary condition used to initiate growth is composed of red tiles as shown in Figure 8(a). As in the construction shown in Figure 1, three boundary tile types with strength-3 bonds can be used to construct this initial condition, or tiles analogous to those in Figure 2 could be used to self-assemble a boundary of size exactly 2^n.

As the assembly grows, as shown in Figure 8(b), the resulting pattern is unique, since at any location where we might try to add a tile, the two notches and the spot always restrict our options so that there is at most one way to add a tile. We can see this by examining just 3 cases: If both notches point up, then we must add a green tile oriented according to the spot. If both notches point down, we must add a red tile (upside down from the one shown in Figure 7) oriented according to the spot. If one notch is down and one is up, then we must use a red tile, but it will only fit if the spot is on the side where the notch points

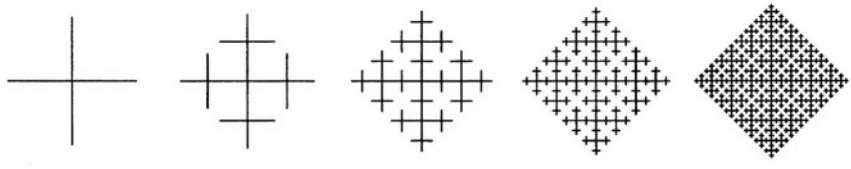


Fig. 9. Successive stages of the "Plus" fractal underlying the pattern of red and green tiles in Figure 8(c,d)

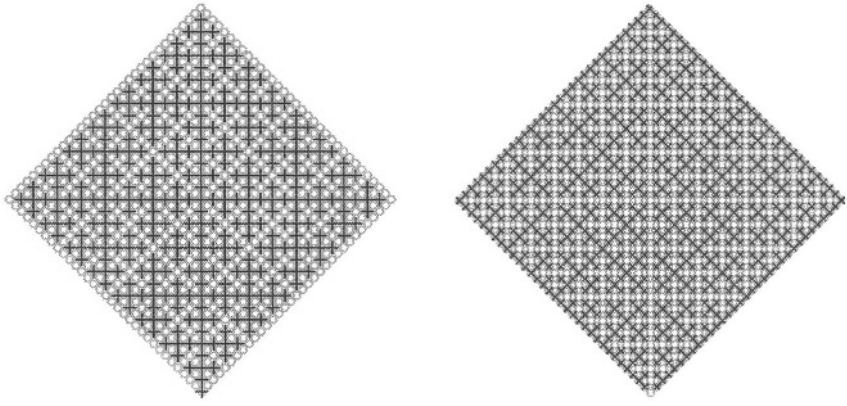


Fig. 10. The green tiles of the pattern in Figure 8(d) happen to be arranged exactly like the "Plus" fractal of Figure 9. Red tiles are also arranged the same way, except the fractal is rotated 45° and centered on an empty point

up. Luckily, we can prove that the spot will indeed always be on this side, as we will show in Section 4.3.

Here we will make some observations about the pattern produced by the tiles, without worrying about proofs. Then in Section 4.3 we will prove that the grown pattern has the self-similar nature being discussed.

Given a fully grown pattern of size $2^n + 1$, we can vertically stretch the rhomboidal array of hexagons from Figure 8(c) into a square array as shown in Figure 8(d) to make the self-similar pattern more apparent. The pattern is the self similar pattern of the "Plus" fractal shown in Figure 9. As shown in Figure 10, if we draw a green + on every green tile in the pattern, then we see exactly the "Plus" fractal, while if we draw a red X on every red tile, we see a simple rearrangement of the same fractal.

In fact, we can draw both the red and the green pattern together on the same tiling, as in Figure 11(a), and in spite of them each covering the entire figure, the red and green fractals do not touch each other at all.

Since these "Plus" fractals have dimension 2, and adjacent tiles in the red pattern are $\sqrt{2}$ times closer than adjacent tiles in the green pattern, there are twice as many red tiles as there are green tiles.

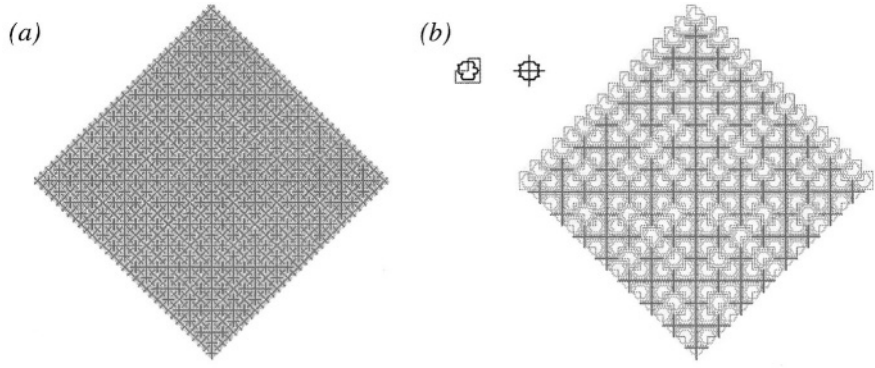


Fig. 11. (a) shows the red and green patterns together. (b) shows the correspondence between the "Plus" fractal and the recursive "L triomino" tiling

On each red tile, instead of drawing a red X, we can draw an L triomino oriented according to the tile, yielding the well-known recursive tiling shown in Figure 11(b).

4.2 The Hadamard Tiles

Now we will modify the red and green tiles to get a set of tiles that can generate a Hadamard matrix. The main modification is just that we will add +1 and –1 markings to the tiles, so we will have a +1 red tile, a –1 red tile, a +1 green tile, and a –1 green tile. The +1 and –1 markings on these tiles are what will form the Hadamard matrix pattern.

We will have the red boundary tiles (corresponding to Figure 8(a)) all carry the +1 marking. The information about whether a tile is marked +1 or –1 will be propagated similarly to how the 0 and 1 markings were propagated in the tiles for the Sierpiński triangle, by labeling edges with "input" and "output" values. Specifically, on each tile we will label the two lower notched edges (the "output" edges) with the tile's main marking, while the two upper notched edges (the "input" edges) will be labeled with compatible inputs. Note that this means we can no longer rotate or flip our tiles, so we will need to explicitly have both orientations of the green tile, and all four orientations of the red tile.

On a red tile, the inputs may be either the same or different, and the tile's main marking always matches the input on the same side as the spot. On a green tile, the inputs are always the same, and the tile's main marking is always the opposite of the inputs. This results in a total of 16 types of red tile and 4 types of green tile.

Note that if we wanted to use square tiles instead of hexagonal tiles, we could eliminate the sides with the spots, and instead communicate the handedness of the spot via the tile to the left of what are now two square tiles touching only at a corner. This breaks some of the symmetry of the tile set (although

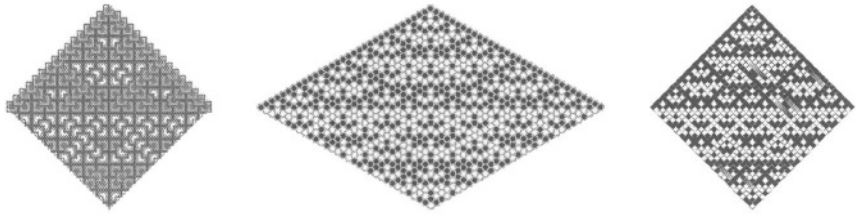


Fig. 12. Three depictions of the self-assembled Hadamard matrix. The leftmost diagram shows how the green/red form of the "Plus"/"L triomino" pattern is related to the Hadamard pattern: Every green tile has the opposite Hadamard color from the tiles above it, while every red tile has the same color as the tile above it in the direction of its axis of symmetry. This surprising relationship between these two fundamental self-similar patterns is the key to how the easily-constructed red and green pattern is used as a stepping stone to the more difficult Hadamard pattern

the marking could also be redundantly communicated on the right as well, to preserve symmetry), but if one needs to use $\tau = 2$ square tiles, it is nice to know that there is no theoretical obstacle.

Figure 12 shows the Hadamard pattern as it is produced on the hexagons together with the red and green patterns, on its own in the original hexagonal setting, and in its square matrix form.

Now we know what to expect when we grow our pattern. Of course, to get a pattern of size exactly 2^n for some given n, one would need to start with a boundary condition of just the right length, which can be accomplished as described in Section 2.

4.3 Proof of Correctness

In this section we prove that the iterative process of tile accretion generates exactly the same pattern as the recursive subdivision process shown in Figure 13.

Since we know from Section 4.1 that at any given position there is only one way a tile can be added to that location, we know that the pattern that grows is the unique pattern satisfying the boundary condition and the edge matching conditions on the tiles. (If there were more than one growable pattern, then the uppermost tile position differing in two such patterns would indicate a place where there is more than one way to add a tile.)

This means that if we can show that the recursive subdivision process always yields an arrangement that is consistent with the growth rules for the notch and spot markings on the edges as well as the Hadamard markings, then it must yield exactly the same arrangement as is generated by the tile accretion process.

To show that the recursive subdivision process never leads to an inconsistency among the tiles, we consider what happens when we subdivide every tile in a consistent pattern X to get a more detailed pattern Y. We will show that if X was consistent, then so is Y.

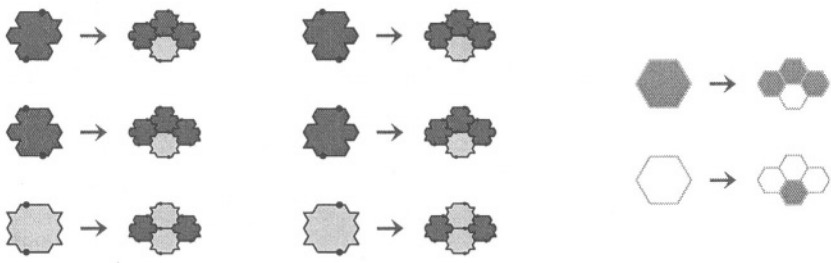


Fig. 13. The left two columns show how red and green hexagons get subdivided into four hexagons. The second column is the mirror image of the first. Every edge of the four small hexagons either is the same in all 6 subdivisions, or takes its notch or spot from a fixed edge of the parent hexagon for all 6 subdivisions. The third column shows how the Hadamard markings are placed on the subdivided hexagons: shaded hexagons represent +1, and white hexagons represent –1. The Hadamard markings coexist with the notch and spot markings, but are shown separately for clarity. If they were shown together, the first two columns would each need to be shown twice: once with the upper Hadamard shadings of column 3, and once with the lower Hadamard shadings of column 3

First we consider the spots. The accretion rule for spots is that the spots must line up on vertically adjacent tiles. Does the subdivision process guarantee that this will be the case throughout any pattern Y which is the result of subdividing a consistent pattern X? There are three cases where tiles in Y need to agree on the spot position: Two vertically adjacent tiles in Y may have come from (A) the same tile in X, (B) vertically adjacent tiles in X, or (C) diagonally adjacent tiles in X. For case (A), we know the spots will agree because we see that they agree in the interior of each individual subdivision rule. For case (B), we know the spots will agree because we see that the alignment of the spots at the top and bottom of each subdivision quadruple match the alignment of the spots at the top and bottom of the parent hexagon, and we know the parent hexagons agreed in X. For case (C), we know the spots will agree because both the lower and upper quadruple have the spot towards the top tile of the lower quadruple, regardless of what quadruples were used.

Next we consider the notches. The accretion rule for notches is that they must match in direction on diagonally adjacent tiles. The subdivision process leads to three cases where two tiles in Y need to agree on the notch direction: (A) the two tiles are in the same quadruple, (B) the two tiles are the top tile of the lower quadruple and a side tile of the upper quadruple, or (C) the two tiles are a side tile of the lower quadruple and the bottom tile of the upper quadruple. For case (A), we know the notches will match because we can see that they match in every possible quadruple. For case (B), we know the notches will agree because regardless of which quadruples are involved, the notch in question will always match the original notch connecting the two parent tiles in X. For case (C), we

know the notches will agree because regardless of which quadruples are involved, the notch will always point down.

Finally we consider the Hadamard markings. The accretion rule for the Hadamard markings is that for a red tile, the marking is copied from the upper left or upper right tile on the side of the spot, while for a green tile, the marking is the opposite of the markings on the upper left and upper right tiles (which happen to have the same markings). We can immediately see that the the Hadamard markings obtained by subdivision obey the accretion rule for the side and bottom tiles of each quadruple, while for the top tile we need to know something about the upper left and upper right neighbor quadruples. What we know about these quadruples is that their side tiles have the same Hadamard marking as their parent in the X tiling. The top tile of the lower quadruple, whose Hadamard marking we are trying to verify, is also marked the same as *its* parent in the X tiling, and in fact it is always the very same tile as its parent. This means that the correct Hadamard marking for its parent in the X tiling is the same as its correct Hadamard marking in the Y tiling, and so since its parent was indeed marked correctly in the X tiling, we know it will be marked correctly in the Y tiling.

Since the spots, notches, and Hadamard markings present after subdivision follow all the rules used for accretion, we see that the subdivision process does indeed yield the growable patterns. If we start with the first tile shown in the subdivision rules, and repeatedly subdivide it to get patterns with more and more tiles, we see that the upper two sides of the resulting array of hexagons match exactly the boundary condition shown in Figure 8. This means that the pattern obtained by repeated subdivision of this tile is exactly the same pattern that grows from that boundary condition. In particular, the Hadamard markings will be exactly those that occur on the pattern grown from the boundary condition. Since it follows from the definition of Hadamard matrices that they are exactly what gets produced by the Hadamard marking subdivision rule shown in the third column of Figure 13, this means that the Hadamard markings on the grown pattern will exactly match the intended Hadamard matrix.

Acknowledgements

M.C. is supported in part by the "Alpha Project" that is funded by a grant from the National Human Genome Research Institute (Grant No. P50 HG02370). P.W.K.R. is supported by a Beckman Postdoctoral Fellowship. E.W. is supported by NSF Career Grant No. 0093486, DARPA BIOCOMP Contract F30602-01-2-0561, and NASA NRA2-37143. Email may be addressed to cook@paradise.caltech.edu.

References

[1] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker. Logic circuits with carbon nanotube transistors. *Science,* 294:1317–1320, November 9 2001.

[2] K. G. Beauchamp. *Walsh Functions and Their Applications.* Academic Press, London, 1975.
[3] M. Boncheva, D. H. Gracias, H. O. Jacobs, and G. M. Whitesides. Biomimetic self-assembly of a functional asymmetrical electronic device. *PNAS,* 99(8):4937–4940, April 16 2002.
[4] A. Carbone and N. C. Seeman. Circuits and programmable self-assembling DNA structures. *PNAS,* 99(20): 12577–12582, October 1 2002.
[5] Q. Cheng and P. M. de Espanes. Resolving two open problems in the self-assembly of squares. Usc computer science technical report #03-793, University of Southern California, 2003.
[6] C.P. Collier, E.W. Wong, M. Belohradsky, F. M. Raymo, J. F. Stoddart, P.J. Kuekes, R. S. Williams, and J. R. Heath. Electronically configurable molecular-based logic gates. *Science,* 285:391–394, 1999.
[7] D.H. Gracias, J. Tien, T.L. Breen, C. Hsu, and G.M. Whitesides. Forming electrical networks in three dimensions by self-assembly. *Science,* 289:1170–1172, Aug. 18, 2000.
[8] B. Grünbaum and G. C. Shephard. *Tilings and Patterns.* W. H. Freeman and Company, New York, 1987.
[9] H. F. Harmuth. Applications of walsh functions in communications. *IEEE Spectrum,* 6:82–91, 1969.
[10] Y. Huang, X. Duan, Y. Cui, L. J. Lauhon, K.-H. Kim, and C.M. Lieber. Logic gates and computation from assembled nanowire building blocks. *Science,* 294:1313–1317, November 9 2001.
[11] H.O. Jacobs, A.R. Tao, A. Schwartz, D.H. Gracias, and G.M. Whitesides. Fabrication of a cylindrical display by patterned assembly. *Science,* 296:323–325, April 12 2000.
[12] M. G. Lagoudakis and T. H. LaBean. 2D DNA self-assembly for satisfiability. In E. Winfree and D. K. Gifford, editors, *DNA Based Computers V,* volume 54 of *DIMACS,* pages 141–154. American Mathematical Society, Providence, Rhode Island, 2000.
[13] C. Mao, T. H. LaBean, J. H. Reif, and N. C. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature,* 407(6803):493–496, 2000.
[14] A. R. Pease, J. O. Jeppesen, J. F. Stoddart, Y. Luo, C. P. Collier, and J. R. Heath. Switching devices based on interlocked molecules. *Acc. Chem. Res.,* 34:433–444, 2001.
[15] S. Pigeon and Y. Bengio. Binary pseudowavelets and applications to bilevel image processing. *Data Compression Conference (DCC '99),* pages 364–373, 1999.
[16] W. Pratt, J. Kane, and H. Andrews. Hadamard transform image coding. *Proceedings of the IEEE,* 57(1):58–68, 1969.
[17] C. Puente-Baliarda, J. Romeu, R. Pous, and A. Cardama. On the behavior of the sierpinski multiband fractal antenna. *IEEE Transactions on Antennas and Propagation,* 46(4):517–524, 1998.
[18] B. H. Robinson and N. C. Seeman. The design of a biochip: A self-assembling molecular-scale memory device. *Protein Engineering,* 1(4):295–300, 1987.
[19] P. W. K. Rothemund. Using lateral capillary forces to compute by self-assembly. *PNAS,* 97:984–989, 2000.
[20] P. W. K. Rothemund and E. Winfree. The program size complexity of self-assembled squares. *Symposium on the Theory of Computing (STOC 2000),* 2000.

[21] J. J. Sylvester. Thoughts on orthogonal matrices, simultaneous sign-successions, and tessellated pavements in two or more colours, with applications to newton's rule, ornamental tile-work, and the theory of numbers. *Phil. Mag.,* 34:461–475, 1867.

[22] S.G. Tzafestas. *Walsh Functions in Signal and Systems Analysis and Design.* Van Nostrand Reinhold, New York, 1985.

[23] J. L. Walsh. A closed set of normal orthogonal functions. *Amer. J. Math,* 45:5–24, 1923.

[24] H. Wang. Proving theorems by pattern recognition. II. *Bell System Technical Journal,* 40:1–42, 1961.

[25] K. A. Williams, P. T. Veenhuizen, B. G. de la Torre, R. Eritja, and C. Dekker. Carbon nanotubes with DNA recognition. *Nature,* 420:761, 2002.

[26] E. Winfree. On the computational power of DNA annealing and ligation. In R. J. Lipton and E. B. Baum, editors, *DNA Based Computers,* volume 27 of *DIMACS,* pages 199–221, Providence, Rhode Island, 1996. American Mathematical Society.

[27] E. Winfree. *Algorithmic Self-Assembly of DNA.* PhD thesis, California Institute of Technology, Computation and Neural Systems Option, 1998.

[28] E. Winfree. Simulations of computing by self-assembly. Technical Report CS-TR:1998.22, Caltech, 1998.

[29] E. Winfree. Algorithmic self-assembly of DNA: Theoretical motivations and 2D assembly experiments. *Journal of Biomolecular Structure & Dynamics,* pages 263–270, 2000. Special issue S2.

[30] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature,* 394:539–544, 1998.

[31] R. Yarlagadda and J. Hershey. *Hadamard Matrix Analysis and Synthesis With Applications to communications and Signal/Image Processing.* Kluwer Academic Publishers, Boston, 1997.

One Dimensional Boundaries for DNA Tile Self-Assembly

Rebecca Schulman, Shaun Lee, Nick Papadakis, and Erik Winfree

Computer Science and Computation & Neural Systems
California Institute of Technology, Pasadena, CA 91125, USA

Abstract. In this paper we report the design and synthesis of DNA molecules (referred to as DNA tiles) with specific binding interactions that guide self-assembly to make one-dimensional assemblies shaped as lines, V's and X's. These DNA tile assemblies have been visualized by atomic force microscopy. The highly-variable distribution of shapes – e.g., the length of the arms of X-shaped assemblies – gives us insight into how the assembly process is occurring. Using stochastic models that simulate addition and dissociation of each type of DNA tile, as well as simplified models that more cleanly examine the generic phenomena, we dissect the contribution of accretion vs aggregation, reversible vs irreversible and seeded vs unseeded assumptions for describing the growth processes. The results suggest strategies for controlling self-assembly to make more uniformly-shaped assemblies.

1 Introduction

Self-assembly - the process by which monomer units come together to form a larger structure according to local, energetic rules - is of great theoretical and practical interest. Most natural self-assembling systems, including crystals, biological membranes, and virus capsids do not permit easy experimental variation of the specificity of binding between units. Control over binding specificity allows the investigation of both the theoretical possibilities of the self-assembly process as well as the practical goal of constructing more complex nano-fabricated patterns.

Control over binding specificity allows the self-assembly process to be *programmed.* The potential of self-assembly programming derives from Wang's investigations of the two-dimensional tiling problem [14], in which he showed that two-dimensional tiling systems can simulate a universal Turing machine [15]. The *(abstract) Tile Assembly Model* (aTAM) [17] is an extension of Wang's tiling systems to include a specific growth process motivated by physical considerations of crystallization. In aTAM, a program is the specification of the tile types, the bond types on their sides, the bond strengths and a threshold value. Assembly begins with a seed tile and proceeds by the non-deterministic addition of tiles at locations where the total strength of all bonds that would be formed is greater than the threshold. The abstract Tile Assembly Model has been shown

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 108–125, 2004.

to be Turing-universal [16, 17], which implies that complex objects can be self-assembled from relatively small numbers of tile types [9]. We therefore say that aTAM supports *algorithmic* self-assembly.

Our goal is to implement the aTAM in chemistry by using DNA tiles – specifically, double crossover (DX) molecules [7] – making use of the complementary base-pairing between its strands to control the strength and specificity of molecular assembly. Each DX molecule consists of two parallel double helices; each strand first participates in one helix, then crosses to the other helix, holding the two helices together. There are four single-stranded *sticky ends* that can hydrogen bond to complementary sticky ends on other DX molecules, allowing fine control of binding specificity. Previous work has shown that the DX molecules R00 and S00 (shown in figure 1) will self-assemble into two-dimensional sheets in which R00 and S00 are arranged periodically in stripes, in agreement with their sticky-end interactions [18]. It remains to be shown experimentally that well-defined algorithmic patterns can result from two-dimensional self-assembly of DNA tiles. Additional forms of control are necessary to achieve this.

As an example of algorithmic self-assembly, consider the set of eight abstract tiles (and their DNA analogs) shown in figure 1 as *boundary tiles* and *rule tiles*[1]. Each DNA tile is designed to have the same bond types, with approximately the same binding strengths, as the corresponding abstract tile; in the abstract model, the **B** bond has twice the strength as the other bond types. See figure 1 for details.

Under the aTAM with the threshold for tile addition set at 2 units, these tiles set up boundary conditions and execute iterated XOR logic to construct Pascal's triangle modulo 2, the discrete analog of the fractal Sierpinski gasket [3]. Self-assembly begins with the *corner tile* RC as the seed tile: Due to the strong bond **B**, first SB, and then RB, can bind repetitively to either side of RC, creating a V-shaped boundary. As soon as a pair of SB tiles have bound to either side of RC, an R11 tile can be added directly above RC, thereby forming two strength-1 bonds and thus achieving the threshold. No other tile could have been added at that location, because all other tile types would form at most one strength-1 bond. As soon as the R11 tile and flanking RB tiles are present, two new locations become available for tile addition; this time, it is only S01 that can make two matching bonds. This process continues forever; at each location, a unique tile may be added, and thus the pattern generated is uniquely defined; an intermediate assembly is shown at the bottom of figure 1.

That the pattern is Pascal's triangle mod 2 derives from the fact that each rule tile corresponds to an entry of the truth table for the XOR function. Conceptually, the inputs to the tile are given in the two sticky ends at the bottom, and the tile's output is (repeated) in both sticky ends at the top. Because of the geometry of the DX molecules we used, and the fact that DNA strands are directed (by convention, from 5' to 3'), tiles in alternating rows have reversed strand orientation. This means that two tiles are required for each line

[1] Two additional tiles, RCxy and S00N, were needed as well for some experiments reported here.

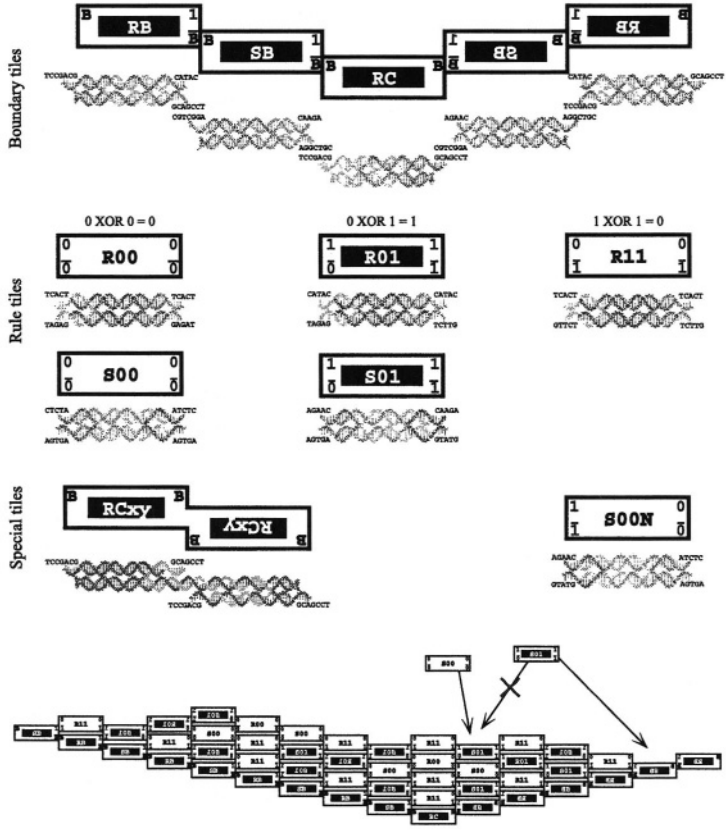


Fig. 1. Abstract tiles, DNA tiles, and their assembly according to the Tile Assembly Model. **Tile types.** Tiles are classified into boundary tiles and rule tiles. Boundary tile RCxy and rule tile S00N, although not part of the assembly process shown at the bottom, were used in experiments reported here. Boundary tiles RC and RCxy are also called corner tiles. A variant of S00N, called S00N-23J, contains hairpin sequences to enhance AFM contrast, as in [18]; it has the same sticky ends as S00N. Each DNA tile is approximately 4 × 12 nm. Abstract tiles may be flipped left-to-right (reflecting a symmetry present in the DNA molecule) as necessary. **Bond types.** Tiles whose names begin with R can bind only to tiles whose names begin with S (because R tiles have 5′ sticky-end overhangs on top, while S tiles have 3′ overhangs, necessitating different sticky end sequences). For the abstract tiles, matching bond types B and $\overline{\mathbf{B}}$ (which are implemented in DNA by GC-rich length-7 sticky ends) have a strength of 2, while the other matching bond types (**0** and $\overline{\mathbf{0}}$, and **1** and $\overline{\mathbf{1}}$, implemented in DNA as length-5 sticky ends) have a strength of 1, in some arbitrary units. **Assembly.** In the bottom half of the figure, tiles are added to a growing assembly either when a **B** bond can be formed or when two weaker bonds can be formed simultaneously. The red X indicates a location where mismatch of the sticky ends prevents tile addition according to the aTAM. Full sequences and sequence design procedures are available at `http://www.dna.caltech.edu/SupplementaryMaterial`

in the truth table. For example, the R01 and S01 tiles have the same semantics (0 XOR 1 = 1), but R01 has 5' overhanging sticky ends on its output (top), while S01 has 3' overhangs. There is no need for an S11 tile, as it does not appear in the Sierpinski pattern.

In order to apply insights about abstract algorithmic self-assembly to real physical systems, we need to understand when a physical system is well-modeled by the aTAM. Initial steps in this direction were achieved in [17], which defined the *kinetic Tile Assembly Model* (kTAM) to include rules of reversible chemistry: any tile may be added at any site at a rate proportional to its concentration as free monomer, and any tile may leave the assembly at a rate exponentially related to the strength of its bonds with the rest of the assembly. In that work, it was argued that physical conditions can be achieved under which the Sierpinski tiles self-assemble correctly with high probability: growth from corner tiles proceeds with a low error rate; growth from rule tiles is very rare; and growth from boundary tiles quickly incorporates a corner tile, and then proceeds with a low error rate.

Although encouraging, the original kTAM model makes assumptions that are not appropriate for some circumstances. Specifically, it assumes (1) that the monomer tile concentration are held constant for the duration of the simulation. Therefore, we call it a *powered* model; in an *unpowered* model, monomer tile concentrations would be depleted as they are used. (2) that assemblies grow only by addition of a single tile at a time, a process called *accretion.* The alternative, *aggregation,* allows two large assemblies to come together and bind to each other. (3) that growth of assemblies is independent of one another, and therefore the fate of a single seed tile can be simulated in isolation. We call this a *(singly-)seeded* model. Assumption (3) is actually a consequence of assumptions (1) and (2); if either monomers are unpowered, or aggregation of assemblies is to be considered, then a *multiply-seeded* model is appropriate. Experiments where DNA molecules are passively assembled in a test tube are more likely to resemble a multiply-seeded, unpowered, aggregation model. Therefore, results from the singly-seeded, powered, accretion kTAM must be carefully interpreted, or – as we do here – enhancements of the original kTAM must be used.

In this work, we address these issues as they apply to the self-assembly of one-dimensional boundaries. Originally intended as a simple step toward demonstrating the Sierpinski pattern experimentally, construction of one-dimensional boundaries has turned out to be an interesting story in its own right. Although the individual DX tiles required for the Sierpinski tile set formed reliably and associated specifically in accord with the programmed interactions, several attempts to create uniformly V- and X-shaped boundaries produced, instead, a high-variance distribution of mostly asymmetrically-shaped assemblies. This turns out to be an excellent test of the various assumptions used in the kTAM. We show, first, that a multiply-seeded, unpowered, accretion variant of the kTAM gives simulation results qualitatively similar to the experimental results. This, however, does not identify which assumptions are valid for our experimental system, nor does it provide understanding. For that, we turn to

three simplified models that test the assumptions of accretion vs aggregation, reversible vs irreversible binding, and singly- vs multiply-seeded growth. Our experimental results are compatible with the reversible and irreversible aggregation models, and incompatible with the irreversible seeded accretion model. Furthermore, our understanding of the models suggests several approaches to solve the problem of creating uniformly-shaped boundaries.

2 Experiments with Seeded and Unseeded Assembly

Early attempts to produce the Sierpinski pattern by mixing the tiles described above were unsuccessful. AFM imaging revealed that some assembly occurred, but it was irregular and difficult to interpret, due to poor AFM resolution at the time. We have since begun a step-by-step process of debugging, testing components of the system one at a time and in simple combinations. During this process resolution has been improved to the point where we regularly can discern individual tiles.

First we tested the boundary tiles in isolation. RB and SB together make what we term a *single-layer* boundary. Figure 2(a) illustrates a typical AFM image of the long filaments formed. Images were difficult to obtain at lower tile concentrations, which precluded identifying individual assemblies. However, it was clear that filaments formed and that they were quite flexible, often forming loops, circles, or coils.

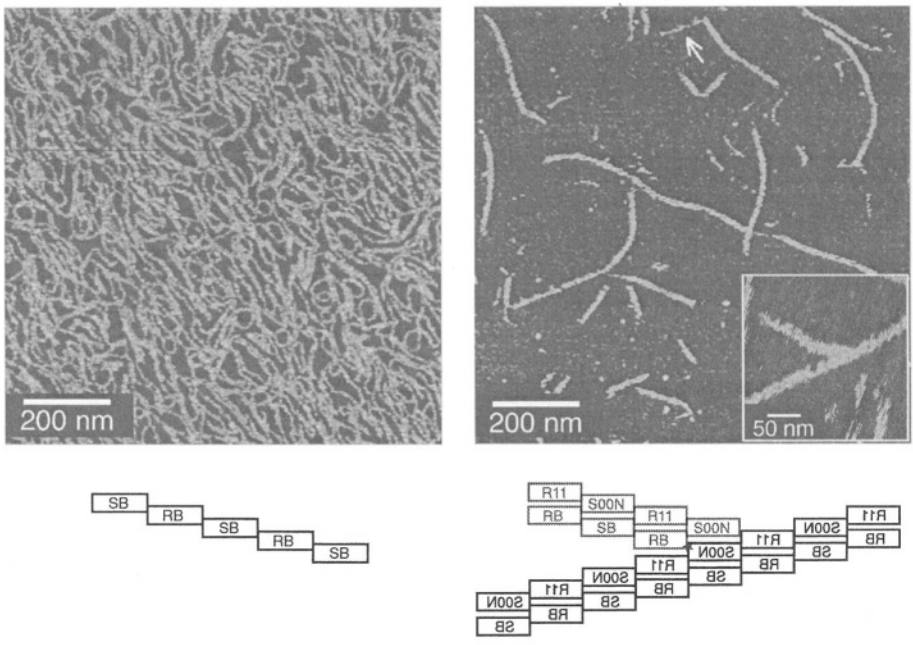
In order to create rigid, straight assemblies, a *double-layer* boundary was formed by adding the R11 and S00N tiles[2] to RB and SB, as shown in figure 2(b). Individual tiles can be distinctly recognized. We also observed long single-layer assemblies, both alone and as *tails* extending from double-layer assemblies (see arrow).

Unfortunately, the addition of small quantities of the RC tile to double-layer boundary did not create shallow V-shaped boundaries suitable for growing large assemblies. There was little evidence of the expected V-shape; images of these samples were essentially indistinguishable from double-layer boundary alone (data not shown). It is not clear whether the few V's that were observed were V-shaped tile arrangements, or simply two boundary assemblies aligned in a V shape by chance, or a boundary assembly folded into a V shape.

The construction of a four-tile assembly consisting of RC, R11, and two SB tiles called the *2x2 V,* verified that the RC tile was binding correctly to its immediate neighbors. Figure 3(a) shows that this construct forms as designed, although at this concentration, 2x2 assemblies appear to associate loosely with each other.

However, when the tiles needed to form the slightly larger V structure shown in figure 3(c) were combined, a *3x3 V* shape was almost never clearly observed. Instead, a great deal of double-layer boundary that may or may not have had a corner tile attached and very occasional shallow V's were visible.

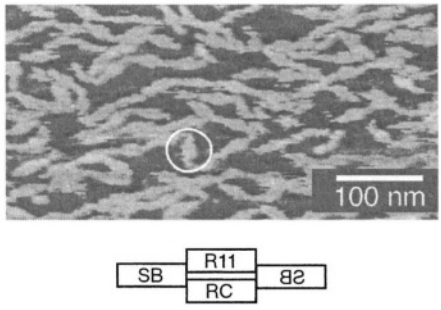
[2] We used S00N here because the combination of R11, S01, RB, and SB tiles might be prone to growing additional layers with mismatches.



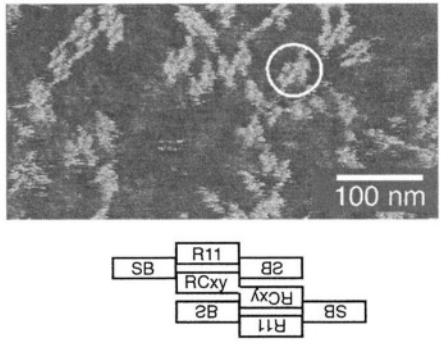
(a) Single-layer boundary at high concentration. RB and SB tiles at 1:1, 2.5μM each tile

(b) Double-layer boundary. Bold portion of the diagram represents the arrangement of the double layer boundary assemblies. The bold tiles combination with the tiles in gray is a hypothesized structure for the motif shown in inset that is seen occasionally. Red star indicates mismatched sticky ends in the putative structure. RB, SB, S00N-23J, R11 at 1:1:1:1, 0.1μM each tile. Diluted 2:1 before imaging

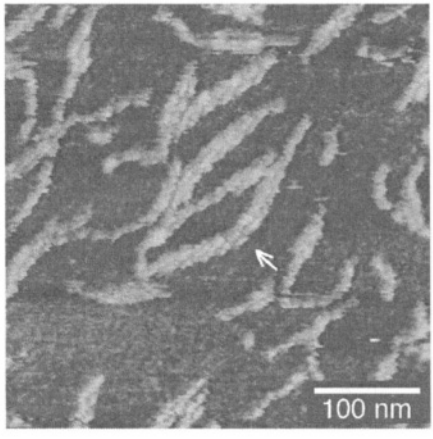
Fig. 2. Boundary assemblies imaged via AFM. **Sample prep:** Oligos were obtained PAGE-purified from Integrated DNA Technologies (www.idtdna.com), and quantified by UV-absorbance. The four strands comprising each tile were mixed at .4μM (each strand) in TAE/Mg^{++} buffer (40mM Tris-Acetate, 1mM Na_2EDTA, 12.5mM Magnesium Acetate) and annealed from 90 to 20 °C at 1 °/min. These tile stocks were mixed at room temperature ($\approx 25°C$) in the specified ratios. Time between mixing and imaging was typically one hour, but varied from 20 min to several days. **AFM imaging:** 5μl of sample was deposited on freshly cleaved muscovite mica (Pella, www.tedpella.com) and allowed to adsorb for 3 min. After optional rinsing with TAE/Mg^{++} buffer, it was imaged under buffer using tapping mode on a Digital Instruments (www.di.com) Nanoscope IIIA with NP-S sharpened silicon nitride tips



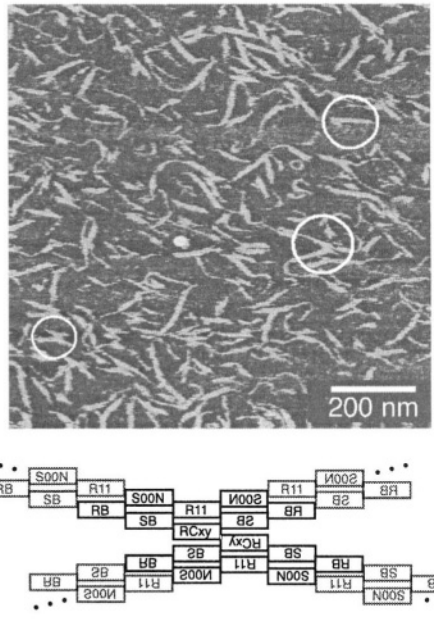
(a) 2x2 V. SB, R11, RC, 2:1:1, RC at 0.1μM



(b) 2x2 X. SB, R11, RCxy, 4:2:1, RCxy at 0.057μM



(c) 3x3 V. RB, SB, S00N-23J, R11, RC, 2:2:2:2:1, RC at 0.044μM. Arrow points to a V-shaped structure that is seen occasionally, and is believed to be the arrangement of tiles shown here in bold and gray.



(d) 3x3 X. Circles indicate examples of 1-, 3-, and 4-armed structures. RB, SB, S00N-23J, R11, RCxy, 4:4:4:2:1, RCxy at 0.027μM

Fig. 3. X and V Structures. In diagrams, black indicates the target assembly (according to stoichiometry); gray indicates other possible polymerization. Experimental methods are as shown in figure 2.

To address the theory that the corner tiles were present, but not visible in some of the previous experiments, a new corner tile that would form assemblies of a distinct X shape was created. This tile, RCxy, resembles an RC tile joined to a second RC tile that has been rotated. The *2x2 X,* a repetition of the 2x2 V experiment with the RCxy tile and adjusted stoichiometries of the other tiles, is shown in figure 3(b). The resulting motif formed without difficulty.

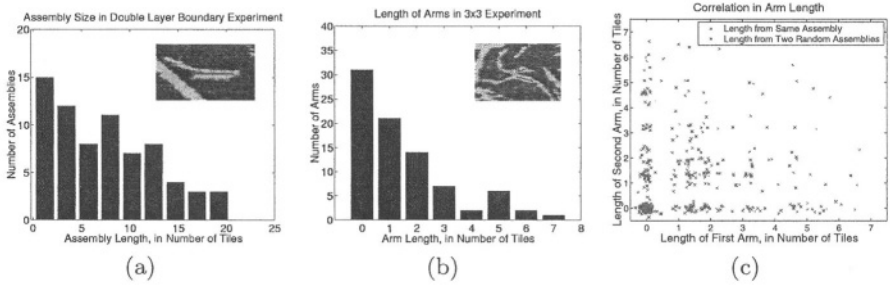


Fig. 4. (a) Measured sizes of double-layer boundary assemblies from an image similar to figure 2(b). (b) Measured sizes of double-layer arms extending from the RCxy corner tile from the image in figure 3(d) (c): A scatter plot of the length of two arms attached to the same corner tile and to different corner tiles, showing that the size of a given arm is independent of the size of the other arms attached to the same corner tile. Arms sizes are calculated from contour sizes of arms as shown in the insets in graphs (a) and (b). The measured contour length of each arm was converted to a length in tiles using the formula 1 tile = 12.5 nm. Simulated measurement noise was added in the scatter plot to avoid exact superposition of datapoints. Single-layer arms, single-layer tails, and assemblies only partially contained in the image were not counted

Finally, the *3x3 X,* shown in figure 3(d), behaved analogously to the 3x3 V. Rather than forming the target assembly, some arms grew relatively long, while others didn't grow at all. The distribution of arm lengths found in an image of double-layer boundary and of 3x3 X is shown in figure 4. Two trends are clear: relative frequency decreases with arm length, and the lengths of arms attached to the same corner tile appear to be statistically uncorrelated.

These experiments confirm that the Sierpinski tiles form structures that reflect their programmed interactions. However, the frequency and shape of the structures that arose were sometimes unexpected.

3 Simulations of Tile Assembly

Our first approach to explaining these results considers a tile-based assembly model that incorporates basic aspects of the physical chemistry of DNA [1]:

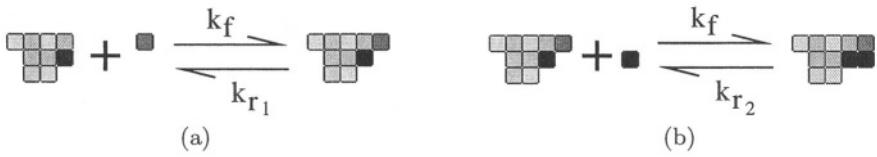


Fig. 5. Growth of tile assemblies based on (a) formation of a single bond, with energy ΔG_1 and reverse rate $k_{r_1} = e^{\Delta G_1/kT}$, and (b) formation of two bonds simultaneously, with energy ΔG_2 and reverse rate $k_{r_2} = e^{\Delta G_2/kT}$

1. The rate of association between tiles, accomplished by the hybridization of their sticky ends, is dominated by their respective concentrations. Therefore, the forward rate constant k_f is identical for all assemblies and all tiles.
2. The rate of dissociation of a tile from an assembly is based on the total free energy, ΔG, of all sticky-end bonds that must be broken. Therefore, the reverse rate constant $k_r = e^{\Delta G/kT}$.

Canonical reactions are illustrated in figure 5. If there were only one reaction involving tile T, assembly A, and assembly A', then

$$\frac{d[A]}{dt} = k_r[A'] - k_f[A][T].$$

The full model allows any tile to add to any assembly at any location, but tiles that do not match their neighbors will have weak bonds and hence dissociate quickly. In principle, the definition of the tile set, including the strengths of all pairwise interactions between tiles, uniquely determines the dynamics of self-assembly. Assembly concentrations evolve over time according to a set of ordinary differential equations, each being a sum of terms similar to the ones shown above. However, as there are an infinite number of different assemblies, an infinite number of ODE's are required. Solving this system explicitly is infeasible in general.

Therefore, we make use of the computationally tractable stochastic kinetic Tile Assembly Model (kTAM) described in [17]. In the original model, a single selected *seed tile* grows into a larger assembly by successive addition or dissociation of single tiles, as described above; it is a seeded, powered, accretion model. Fortunately, at steady-state (if it exists), the probability of observing a particular assembly in the simulation is proportional to the concentration of that same assembly in the full model at equilibrium, so long as in both cases the steady-state monomer tile concentrations are the same [12].

In order to use this model to simulate our experimental systems – where the set of tile types and their total initial concentrations are known, but the equilibrium concentration is not known – two enhancements of kTAM are necessary, which we call the *multiply-seeded, unpowered, accretion* kTAM (multi-kTAM). First, multiple assemblies are grown simultaneously; and second, the concentration of each monomer tile (shared by all assembly growth processes) is depleted with each tile addition and restored with each tile dissociation. Thus, there are

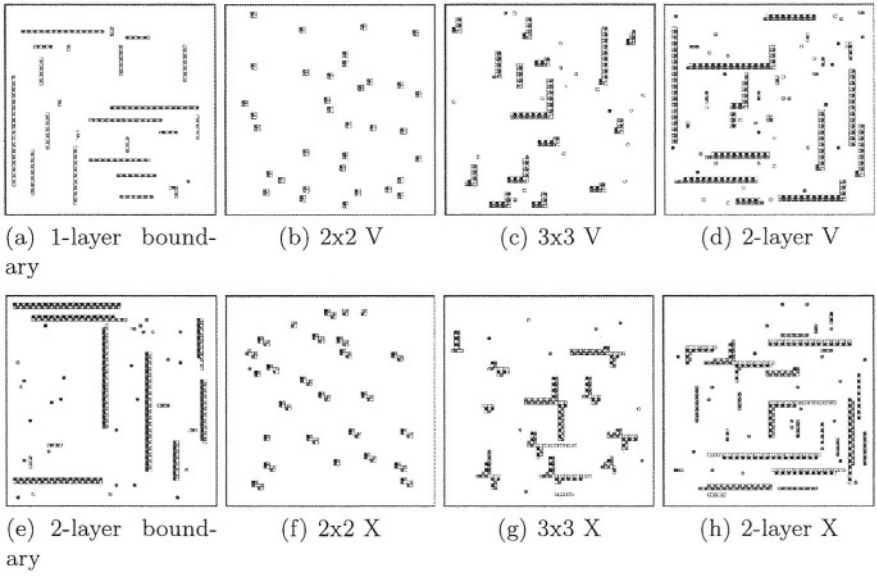


Fig. 6. A selected sample of assemblies in the steady-state distribution of multi-kTAM simulations for each experiment shown in figures 2 and 3. Initial monomer tile concentrations were chosen exactly in correspondence to the experimental conditions, with the exception that concentration for 2-layer V and 2-layer X simulations were as in the corresponding 3x3 experiment, but with RC diluted 100:1. Each simulation was started with $N_A = 100$ seed tiles of each type, each with $C_A = [monomer]/1000$. Tile binding strengths ($\Delta G = 7.2$ kcal/mol for the **0** and **1** bond types) were chosen based on length-5 sticky ends at 25°C, as in [17], with the **B** bonds between boundary tiles *RB* and *SB* being treated as twice as strong as the bonds between other tiles. Mismatched bonds between tiles with non-complementary sticky ends were assigned $\Delta G = 0$. Simulation code and parameter files are available at `http://www.dna.caltech.edu/SupplementaryMaterial`

two new parameters to the model: how many assemblies to simulate, N_A, and their "effective concentration", C_A, determining how much the global tile concentration changes with each monomer association or dissociation. Unfortunately, the choice of N_A and C_A can bias the steady-state distribution in the simulation, and it is at this point unclear how to optimally choose those parameters in order for the simulation to accurately reproduce the equilibrium distribution of assemblies defined by the full model. Therefore, although we believe we chose reasonable parameters for our simulation, the results of the multi-kTAM simulations must be considered qualitative until better understanding of the model is achieved.

We ran the multi-kTAM simulation for tile sets modeling each of the six experiments described in the previous section. The results, depicted qualitatively in figure 6, reproduce main features of AFM qualitatively: whereas the 1-layer, 2-layer, and V and X 2x2 tile sets all form the desired structures, but the 3x3

tile set polymerizes into assemblies with predominantly just one or two arms. At low simulation temperatures, the defect shown in figure 2(b) (inset), also occurs in the simulations.

This gives us confidence that the unexpected features we observe in the AFM images are not necessarily due to ill-formed tiles, bad DNA, old chemicals, or unknown physical or chemical effects, but rather, they may be due solely to the processes incorporated into our model. This is encouraging, because it implies that our model may provide the necessary insights required to fix the problem. However, because of the dependence on the parameters N_A and C_A, we are not confident that the distributions resulting from multi-kTAM simulations are the correct predictions of the general physical model of tile-based assembly. Furthermore, the simulations do not give us a clear intuitive understanding of *why* the model reproduces these effects. For that, we turn to simplified models that can be analyzed exactly.

4 A Theory of Boundary Tile Assembly

Our experimental and simulation results can be summarized in the observation that small, well-defined assemblies form as expected, and that experiments in which the tiles could polymerize (*i.e.,* form arbitrarily long chains) produced a distribution of assemblies in which the target was a rarity.

Therefore, with the goal of understanding the general features of boundary formation, we discuss a class of models that further idealizes the kTAM model and are simple enough that they can be analyzed and reasoned about intuitively.

Specifically, three models will be considered: (a) an irreversible seeded process, (b) reversible aggregation and accretion systems at equilibrium, and (c) an irreversible aggregation process. The experimentally measured arm lengths may seem to be more consistent with models (b) and (c).

In what follows, we reduce the formation of DNA tile boundaries to the formation of one dimensional heterogeneous polymers containing two types of monomers: the corner tile and the generic boundary tile. A corner tile may only occur once in the assembly and can bind up to four boundary tiles. Boundary tiles may attach to corner tiles and polymerize linearly. We refer to the boundary tile species as B and the corner tile species as C. Assemblies may consist of a line of boundary tiles, or a corner tile connected to four such lines (in this context called arms) forming an X shape. A boundary assembly is referred to as B_n, $n > 0$ and a four armed assembly that contains a corner tile and arms of lengths m_1, m_2, m_3, and m_4 (with $m_i \geq 0$) is referred to as $CB_{m_1}B_{m_2}B_{m_3}B_{m_4}$. All the reactions that we consider involve growth or shrinkage of one of the four assemblies attached to the corner independent of the length of the others. For simplicity we will discuss models with one arm per corner tile, with the exception of the graphical depiction of assemblies produced by our various models (figures 8(d), 8(h), and 8(1)), and a discussion of the difficulty of creating an assembly where all arms are long. In all models discussed here, the dynamics of systems with X shaped corner tiles are identical to those of a one armed system, where each corner tile

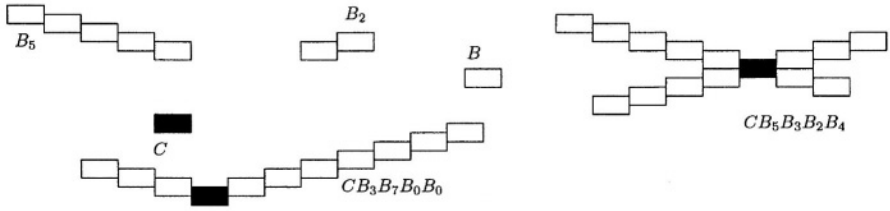


Fig. 7. Types of species in our theoretical models of boundary assembly include corner tiles attached to several boundary assemblies, and boundary and corner tiles alone

in the one armed system is replaced by a set of four corner tiles. The arm lengths of this group of four tiles corresponds to the four arms of a single X.

4.1 Seeded Irreversible Tile Assembly by Accretion

The original kTAM model assumes that growth always begins from a seed tile. This supposition stems from the fact that the formation of a bond between two rule tiles is not energetically favored unless two such bonds can form at once. The assumption that growth begins from a seed nucleus is standard in crystal growth, where crystal formation is believed to be primarily governed by accretion.

Seeded irreversible accretion can be modeled by a single reaction:

$$B + CB_n \rightarrow CB_{n+1} \qquad n \geq 0 \tag{1}$$

Assembly formation through this reaction causes each boundary tile to randomly attach to a corner assembly, completing when every boundary tile has been attached to an assembly. The result is that, considered in isolation, the length of an arm is binomially distributed. Where L is a random variable representing the length of a single arm, n is a length of interest, c the number of corner tiles, and b the number of boundary tiles, and $p = \frac{1}{c}$,

$$Pr(L = n) = \binom{b}{n} p^n (1-p)^{b-n}$$

The length of the arms are approximately independent, so that the total distribution looks like a binomial distribution (figure 8(b); *ie* the expected number of arms with length n is approximately $c * Pr(L = n)$.

The variance of this distribution becomes small when the ratio of boundary to corner tiles and the total number of molecules grow large. Therefore, if this process were responsible for boundary assembly formation, corner and boundary tile experiments would reveal many evenly sized assemblies uniformly attached to corner tiles (figure 8(d)).

4.2 Reversible Tile Assembly

Another possibility was that the tile assemblies observed experimentally were at conditions approximating equilibrium. In this case, boundary tiles would be in

equilibrium not only with corner assemblies but possibly also with assemblies made up solely of boundary tiles. The equilibrium conditions for two types of reaction systems are considered here: an accretion system, where only single tiles join assemblies with or without a corner tile, and an aggregation system, where boundary assemblies as well as corner assemblies can form, and boundary assemblies may interact as boundary tiles do. The reactions for the two models are (for $n > 0, m \geq 0$):

Accretion	Aggregation
$B + CB_m \rightleftharpoons CB_{m+1}$	$B_n + CB_m \rightleftharpoons CB_{m+n}$
$B + B_m \rightleftharpoons B_{m+1}$	$B_n + B_m \rightleftharpoons B_{n+m}$

The corner and boundary tiles have the same on and off rates, so we assigned each reaction a common forward rate constant k_{on}, backward rate constant k_{off}, and equilibrium constant $K_{\text{eq}} = \frac{k_{\text{on}}}{k_{\text{off}}}$.

These reactions are simple enough to calculate the equilibrium values in closed form. The accretion model has the same equilibrium as the aggregation model. [3] At equilibrium in the accretion model, for all $m, p \geq 0$ we have:

$$\frac{[B_{m+n}C]}{[B_mC][B_n]} = K_{\text{eq}} \qquad \frac{[B_{m+n}]}{[B_m][B_n]} = K_{\text{eq}} \tag{2}$$

Solving these gives

$$[B_n] = [B](K_{\text{eq}}[B])^{n-1} \qquad [B_mC] = [C](K_{\text{eq}}[B])^m \tag{3}$$

These equation predict that at equilibrium the sizes of assemblies should be geometrically distributed with parameter $(K_{\text{eq}}[B])$, where $[B]$ is the concentration of lone boundary tiles at equilibrium. This is a well known fact about polymerization [4, 5].

The equilibrium solution is entirely symmetric with respect to the four arms of an X shape. Thus for a four armed assembly the concentration of an assembly type is

$$[CB_{m_1}B_{m_2}B_{m_3}B_{m_4}] = [C](K_{eq}[B])^{(m_1+m_2+m_3+m_4)}$$

This implies that the total number of tiles connected to a corner X tile is geometrically distributed. Because tiles are distributed independently across each of the four arms of a corner X tile, we can conclude that corner tiles with

[3] The kinetics of boundary formation in a reversible accretion process are much slower than for a reversible aggregation process. Seeded reversible accretion growth also gives a geometric distribution of tiles at equilibrium. However, in an accretion model, the concentration of the single boundary species $[B]$ is much greater than in the aggregation model, where most boundary tiles can react with each other to form boundary assemblies in addition to connecting to corner tile assemblies. Thus the geometric distribution parameter $K_{eq}[B]$ for the equilibrium concentrations is much closer to one in the accretion case. In accretion, we see much more even but still geometric distributions of corner assembly sizes.

four long arms are rare, for any reasonable definition of "long"[4]: the probability that all four arms are long is the fourth power of the probability that any single arm is long. Similarly, for corner V tiles, the probability of two long arms is the square of the probability of one. The arm length distribution for of a reversible process that approaches equilibrium is shown in figure 8(f). Most corner tiles do not have more than one or two nontrivial arms (figure 8(h)).

4.3 Irreversible Tile Assembly by Aggregation

Irreversible tile assembly would appear to be a limiting case of the reversible aggregation model, where binding reactions become so biased toward the forward reaction that they are essentially irreversible. The results (shown in figures 8(j), 8(l)) of a simulated irreversible aggregation process are similar to the reversible case, but have a more accentuated fraction of length-0 arms, and the distribution has longer tails. In irreversible aggregation tiles "freeze" when they are first attached to a corner tile, rather than finding a final equilibrium state. Therefore, the results of the irreversible tile assembly model are likely to resemble a kinetic intermediate rather than the equilibrium of a set of tiles where binding between tiles is very strong.

5 Discussion

As part of our work on the creation of complex self-assembled structures, we have designed and experimentally verified the binding specificities of many different DNA double crossover molecules. We have been able to form specific structures with these tiles, but not yet control their polymerization. Moreover, the kinetic tile assembly model's predictions correspond well with what is seen experimentally. We can further simplify this model to explore extremes in the behavior of tiles, and to provide simpler explanations for our experimental results.

The experimental statistics shown here represent an initial effort at quantification, and are not yet conclusive. Several sources of error may be present: A small number of images were analyzed, selected for clarity; they may not be representative of typical results. Sample preparation techniques, variations in the adhesiveness of the mica, or AFM tip interactions might affect the observed distribution of assemblies. The size or shape of an assembly might further affect its ability to adsorb.

Definitive and more substantial conclusions would require additional samples to be analyzed; in particular, experiments that vary the stoichiometric ratio of boundary to corner tiles, imaged with resolution than can distinguish between a boundary assembly and a corner assembly with one arm, would help discriminate between the growth models. It would also be instructive to compare to

[4] For example, one could say an arm is "long" if it has at least half the average number of tiles.

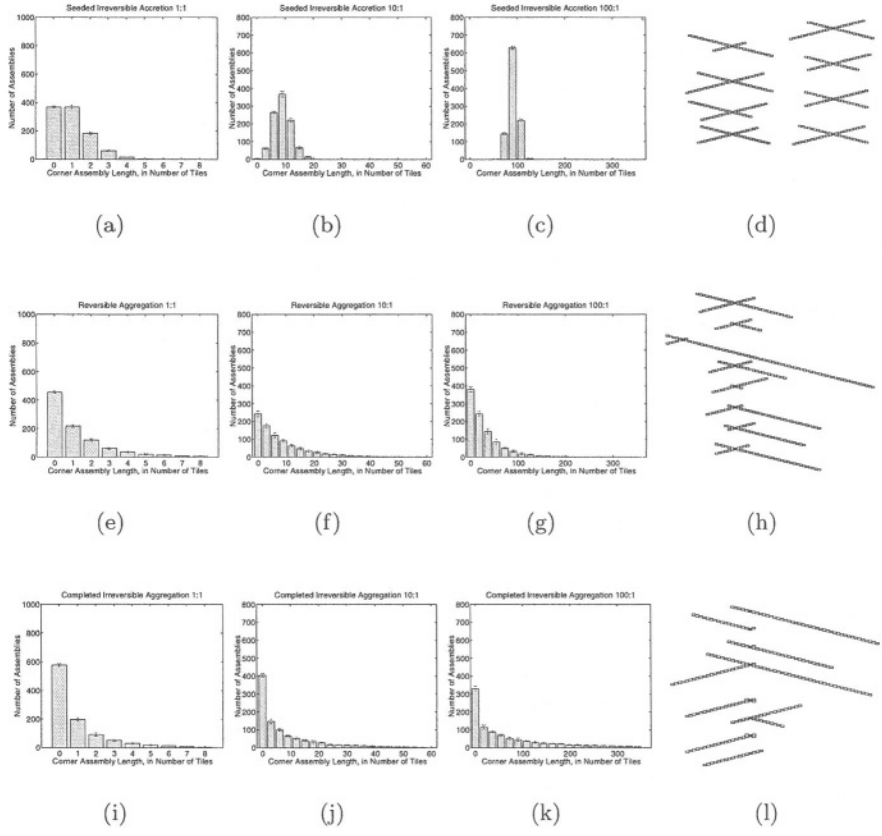


Fig. 8. Distributions of the size of assemblies attached to corner tiles for, (8(a), 8(b), 8(c)) the seeded irreversible accretion model at completion, (8(e), 8(f), 8(g)) reversible aggregation model close to equilibrium, and (8(i), 8(j), 8(k)) the irreversible aggregation model at completion. The column shows results for 1:1, 10:1, and 100:1 stoichiometric ratios of boundary tiles to corner tiles. The last column (8(d), 8(h), 8(l)) gives a visual depiction of eight assemblies randomly chosen from each set of results, for each of the three models when a 10:1 stoichiometric ratio is used. All simulations used $c = 1{,}000$ corner tiles and $b = 1{,}000$, 10,000 or 100,000 corner tiles. For irreversible models, at each step two assemblies were chosen from the collection, and if a reaction between them was allowed by the model being simulated, the assemblies were replaced by the reaction product. For the reversible model, each possible association or dissociation reaction between two assemblies was chosen randomly, weighted by rates K_{on} and K_{off} where $\frac{K_{on}b}{K_{off}[B]} = K_{eq}$. Assuming the concentration of boundary tiles was $[B] = 0.1\mu$M, the equilibrium coefficient in the reversible reaction was $K_{eq} = 1$ nM. Simulation code and scripts for these figures are available at `http://www.dna.caltech.edu/SupplementaryMaterial`

boundary formation at colder temperatures, which should be an essentially irreversible process, and to boundaries formed by slow annealing, which should yield the distribution of a reversible process.

Furthermore, the experimental results shown here are insufficient for extracting thermodynamic parameters (ΔG, or equivalently, K_{eq}) for tile binding events between our specific molecules. Are the parameters used in the simulations reasonable? Based on generic nearest-neighbor parameters for base stacking within a duplex [10], coaxial base stacking at nick sites [13], and dangling ends [2], we estimate[5] rule tile sticky ends to bind with K_{eq} between 2 and 200 μM, and boundary tile sticky ends to bind with K_{eq} between 10 and 100 nM at 25°C. The multi-kTAM simulations used 6 μM and 34 pM respectively, while the reversible aggregation simulations used 1 nM for the boundary tiles. Thermodynamic parameters for our DNA tiles should be measured experimentally.

Despite these difficulties, the results obtained here, when compared to the models, support qualitative conclusions, because the generic shape of the assembly shape distributions is not dependent upon the exact parameter values. Among the simplified models, the predictions of the irreversible aggregation (figure 8(i)) and reversible aggregation 8(e)) are closer to the experimental observations (figure 4) than the irreversible accretion model (figures 8(a)). These models also seem *a priori* more plausible than the irreversible accretion model, because boundary tiles have no mechanism to prevent aggregation of boundary assemblies prior to attachment to a corner assembly. Unfortunately, it will be difficult to build corner assemblies with long arms using irreversible or reversible aggregation processes.

Thus, the models suggest that one difficulty in creating large, uniformly-sized V and X assemblies may be the ability of boundary assemblies to form in the absence of corner tiles. In this case, neither increasing the concentration of boundary tiles in our reactions, nor increasing the binding strength between boundary tiles, will significantly increase the concentration of large, uniformly-sized corner assemblies. This also means that using ligase to lock the tiles together will be ineffective for making large, uniformly-size corner assemblies.

On the other hand, the seeded accretion model shows that if growth can be constrained to occur from the corner tile only, then large, relatively uniformly-sized corner assemblies will result. We are designing a new set of boundary tiles for a Sierpinski triangle that, even in an aggregation model, dramatically reduce the ability of boundary tiles to spontaneously assemble in the absence of a corner nucleus [11]. Preventing such spontaneous growth is also likely to reduce the effects of stoichiometry poisoning, discussed below.

Several other factors, other than the ones discussed above, may also be playing a role in shaping the experimentally observed distributions. For example, malformed DNA tiles could terminate growth, leading to truncated distributions. More subtle is the effect of stoichiometry poisoning. Ideally, the concentrations

[5] The wide range of the estimates is due to sequence dependence of the parameters (giving different sticky ends different K_{eq}'s), as well as being due to uncertainty about the reported parameters.

of RB and SB tiles in our experiments are equal, but in practice small pipetting errors cause differences in the amount of each type of tile present. These slight differences can greatly influence the size of assemblies that form, since in a polymerization reaction involving alternating monomer types, the more unequal the two concentrations are, the smaller the polymers become on average [6]. In the case where [RB] > [SB], most assemblies end up in the form RB-(SB-RB)$_n$. These assemblies cannot combine, so assembly slows down or halts. While individual pipetting steps are accurate to within 2 percent, multiple pipetting steps might increase the total error to as much as 5 or 10 percent. For a concentration difference between RB and SB tiles in this range, assemblies larger than 10 or 20 tiles would be rare.[6]

Previous work on DNA computation by self-assembly made use of DNA triple-crossover tiles that assemble into a seeded 2-layer one-dimensional array to compute a 4-bit cumulative XOR operation [8]. Attempts to extend that work to to longer inputs would likely be governed by the same principles discussed here.

The ultimate goal of one-dimensional boundary formation, in the context of algorithmic self-assembly, is to provide input for subsequent two-dimensional growth. However, the relationship between boundary formation and two dimensional crystal growth is not well understood. It is possible, for example, that problems with boundary formation would be ameliorated if rule tiles were simultaneously filling in the space between two arms on a V or X, thus reducing the troublesome statistical independence.

Acknowledgements

We would like to thank Bernie Yurke for pointing out the effects of slight differences in stoichiometry, and for his very descriptive term “stoichiometry poisoning”. Paul Rothemund, Rizal Hariadi, and other members of the DNA Lab provided helpful hints and stimulating conversation. This work was supported by NSF CAREER Grant No. 0093486, DARPA BioComputation Contract F30602-01-2-0561, NASA NRA2-37143, and GenTel.

References

[1] V. A. Bloomfield, D.M. Crothers, and I. Tinoco, Jr. *Nucleic Acids: Structures, Properties, and Functions.* University Science Books, 2000.

[6] As the concentrations of the two monomer types become more uneven, the distribution is limited by the ratio $r < 1$ rather than by the equilibrium constant between the monomers. Under irreversible binding, the size distribution is

$$[\text{RB-(SB-RB)}_n] = r^{(2n+1)}\frac{(1-r)^2}{1+r}. \quad (4)$$

In the case of 5-10 percent pipetting error $r \approx 0.9$.

[2] S. Bommariro, N. Peyret, and J. SantaLucia, Jr. Thermodynamic parameters for DNA sequences with dangling ends. *Nucleic Acids Research,* 28(9): 1929–1934, 2000.

[3] B. A. Bondarenko. *Generalized Pascal Triangles and Pyramids, Their Fractals, Graphs and Applications.* The Fibonacci Association, 1993. Translated from the Russion and edited by Richard C. Bollinger.

[4] P. J. Flory. Molecular size distribution in linear condensation polymers. *Journal of the American Chemical Society,* 58:1877, 1936.

[5] P. J. Flory. Fundamental principles of condensation polymerization. *Chemical Reviews,* 39:137, 1946.

[6] P. J. Flory. *Principles of Polymer Chemistry.* Cornell University Press, 1953.

[7] T.-J. Fu and N. C. Seeman. DNA double-crossover molecules. *Biochemistry,* 32:3211–3220, 1993.

[8] C. Mao, T. H. LaBean, J.H. Reif, and N.C. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature,* 407(6803):493–496, 2000.

[9] P. W. K. Rothemund and E. Winfree. The program-size complexity of self-assembled squares. In *Symposium on Theory of Computing (STOC).* ACM, May21–23 2000.

[10] J. SantaLucia, Jr. A unified view of polymer, dumbbell, and oligonucleotide dna nearest-neighbor thermodynamics. *Proc. Nat. Acad. Sci. USA,* 95:1460–1465, 1998.

[11] R. Schulman and E. Winfree. Controlling nucleation rates in algorithmic self-assembly. In preparation.

[12] R. Schulman and E. Winfree. Relationships among tile assembly models. In preparation.

[13] V. A. Vasiliskov, D. V. Prokopenko, and A. D. Mirzabekov. Parallel multiplex thermodynamic analysis of coaxial base stacking in DNA duplexes by oligodeoxyribonucleotide microchips. *Nucleic Acids Research,* 29(11):2303–2313, 2001.

[14] H. Wang. Proving theorems by pattern recognition. II. *Bell System Technical Journal,* 40:1–42, 1961.

[15] H. Wang. Dominoes and the AEA case of the decision problem. In J. Fox, editor, *Proceedings of the Symposium on the Mathematical Theory of Automata,* pages 23–55, Brooklyn, New York, 1963. Polytechnic Press.

[16] E. Winfree. On the computational power of DNA annealing and ligation. In R.J. Lipton and E. B. Baum, editors, *DNA Based Computers: DIMACS Workshop, April 4, 1995,* volume 27, pages 199–221, Providence, RI, 1996. American Mathematical Society.

[17] E. Winfree. Simulations of computing by self-assembly. Technical Report CS-TR: 1998.22, Caltech, 1998. Originally appeared in the preliminary proceedings of the 4th DIMACS Meeting on DNA Based Computers, held at the University of Pennsylvania, June 16-19, 1998.

[18] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature,* 394:539–544, 1998.

Proofreading Tile Sets: Error Correction for Algorithmic Self-Assembly

Erik Winfree and Renat Bekbolatov

Computer Science and Computation & Neural Systems
California Institute of Technology
Pasadena, CA 911125, USA

Abstract. For robust molecular implementation of tile-based algorithmic self-assembly, methods for reducing errors must be developed. Previous studies suggested that by control of physical conditions, such as temperature and the concentration of tiles, errors (ε) can be reduced to an arbitrarily low rate – but at the cost of reduced speed (r) for the self-assembly process. For tile sets directly implementing blocked cellular automata, it was shown that $r \approx \beta\varepsilon^2$ was optimal. Here, we show that an improved construction, which we refer to as proofreading tile sets, can in principle exploit the cooperativity of tile assembly reactions to dramatically improve the scaling behavior to $r \approx \beta\varepsilon$ and better. This suggests that existing DNA-based molecular tile approaches may be improved to produce macroscopic algorithmic crystals with few errors. Generalizations and limitations of the proofreading tile set construction are discussed.

1 Introduction

The experimental demonstration that DNA can be used to encode and process information [1] has stimulated interest in how biomolecular processes can be programmed to carry out logical algorithms. Algorithmic self-assembly of DNA tiles has been proposed as the basis for parallel computation of solutions to hard combinatorial problems [27, 30, 18, 12] and for bottom-up nanofabrication of complex structures that can be specified by simple rules [23, 20, 8].

Understanding of this approach relies upon an abstract model of the growth process, known as the abstract Tile Assembly Model (aTAM). As in Wang's Tiling Problem [25, 26], a tile system consists of a finite set of square tiles with a label on each side. For simplicity, tiles cannot be rotated. The aTAM augments these tiles with a *strength function* that specifies how tightly tiles stick to each other when the labels on touching sides match (referred to as a *bond*). This motivates a *growth rule:* starting with a specified *seed tile,* a new tile may be added at any position where the total strength of all newly-formed bonds exceeds a threshold, τ. This rule is intrinsically asynchronous and non-deterministic: only certain tile sets will produce a uniquely-defined structure. The aTAM is illustrated in figure lab, using a tile set we call the Sierpinski tiles; at $\tau = 2$, the growth results [28] in an infinite Sierpinski triangle pattern [5].

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 126–144, 2004.

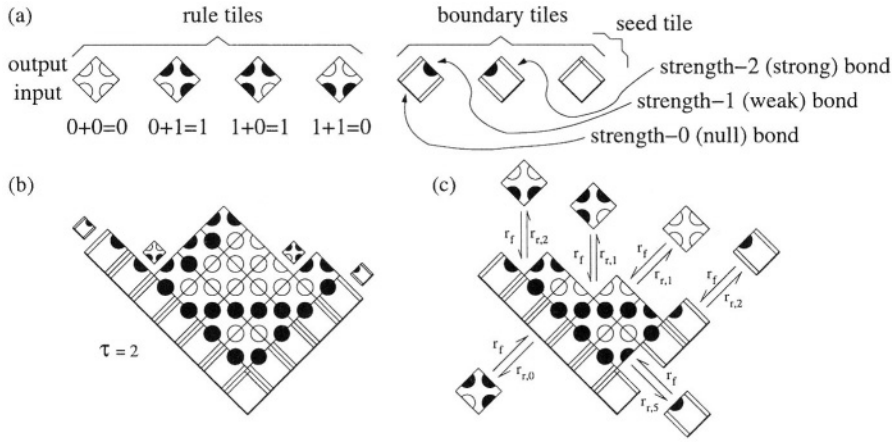


Fig. 1. **(a)** The seven Sierpinski tiles include four *rule tiles* implementing the XOR logic and three *boundary tiles,* including the *corner tile* which is used as the *seed tile.* Whereas all sides of the rule tiles form strength-1 (weak) bonds, the boundary tiles also make use of strength-2 (strong) bonds and strength-0 (null) bonds. Strong bonds are drawn as double lines, and null bonds are drawn as thick lines. This construction can be trivially generalized to implement an arbitrary BCA with more than two symbols. **(b)** Growth of the Sierpinski tiles from the seed tile according to the aTAM at $\tau = 2$. The small tiles indicate the (only) four sites where growth can occur. At each location, there is a unique tile that may be added, and a unique pattern results, **(c)** Rates for tile addition and tile dissociation in the kTAM: r_f is the forward rate for association of any tile at any site, and $r_{r,b}$ is the reverse rate for dissociation of a tile that makes bonds with total strength b. All and only single monomer tile associations and dissociation events are considered in the kTAM; a representative selection is shown here

The Sierpinski rule tiles implement a replacement rule $(x, y) \rightarrow (z, z)$ where $x, y \in \{0, 1\}$ and $z = (x + y) \mod 2$; each tile has some x and some y on the bottom two sides and the corresponding z on both top sides. This is an instance of a one-dimensional *blocked cellular automaton* (BCA): an initial string, the bottom layer, is transformed for each subsequent layer by partitioning the string into pairs (alternately odd/even-indexed and even/odd-indexed elements) and replacing each pair according to a rule $(x, y) \rightarrow (f_1(x, y), f_2(x, y))$, where all values are in an alphabet, *e.g.,* {0,1,..., N}. Generalizing the Sierpinski rule tiles to rule tiles whose inputs and outputs may take on $(N + 1)$ values yields a direct implementation for any chosen BCA: one rule tile is used for each of $(N + 1)^2$ possible input pairs, and the initial conditions for the computation are given using boundary tiles with strength-2 bonds that grow from the seed tile. We call this the *direct tile set* for a given BCA. Since BCA are capable of Turing-universal computation, this implies that tile-based self-assembly provides a natural logical basis for programmable self-assembly processes with arbitrary complexity. We now need to find a physical process, such as DNA tile assembly, wherein self-assembly occurs (at least approximately) according to the aTAM rules at $\tau = 2$, using tiles with bond strengths restricted to 0, 1, and 2.

Tile-based self-assembly of two-dimensional periodic structures has been successfully demonstrated experimentally using a variety of molecular implementations of DNA tiles [29, 11, 14, 31]. The key principle is that each DNA molecule has *four* short single-stranded regions, known as *sticky ends,* which direct how the DNA tiles bind to each other: sticky ends with complementary sequences can form a thermodynamically favorable double-helix, as shown in figure 2 for DNA tiles made from DNA double-crossover molecules [9]. Attempts at *algorithmic* self-assembly in one dimension [13] and in two dimensions [19] have also proven successful, but with two limitations: (1) incorrect tiles are incorporated into the growing structure with error rates ranging from 1% to 10%, and (2) spurious nucleation (not involving the seed tile) results in many structures that perform the wrong computation and thus produce an undesired structure.

Our concern in this paper is how to control growth errors.[1] We can consider four approaches to reducing growth errors.

Logical Error Correction. Accept an intrinsic error rate ε, and design a larger tile set that contains logic to detect and correct errors. In principle, this should be possible by making use of one-dimensional fault-tolerant cellular automata [10], but it is likely to be extremely complicated.

Optimized Physical Conditions. Study how physical conditions, such as temperature, tile concentrations, and buffer conditions, determine the error rate, and optimize them to obtain the best performance. As was shown in [28], this approach is promising, but it is likely to require extremely slow growth conditions.

New Molecular Mechanisms. Devise new physical mechanisms, such as, for example, more complicated molecular implementation of tiles with latches and switches. However, new structural motifs are difficult to design and characterize, so this approach must be considered with caution.

Exploiting Cooperative Binding. While retaining the original molecular tile design, redesign the original tile set to exploit physical mechanisms already inherent in the self-assembly process. This combined approach, which relies both on logical aspects of the tile set and on physical aspects of the assembly process, is explored here with dramatic benefits.

2 The Kinetic Tile Assembly Model: Error vs Growth Rates

Both growth errors and nucleation errors can be understood in terms of an extension of the aTAM to include rates both for tiles associating to and for tiles dissociating from the growing crystal (figure 1c); this model is known as the kinetic Tile Assembly Model (kTAM) [28]. The on-rates and off-rates can be chosen according to the principles of DNA hybridization [4], as illustrated in figure 2 for tiles implemented as DNA double-crossover (DX) molecules [9]. The

[1] Nucleation errors will be treated in an upcoming paper [22].

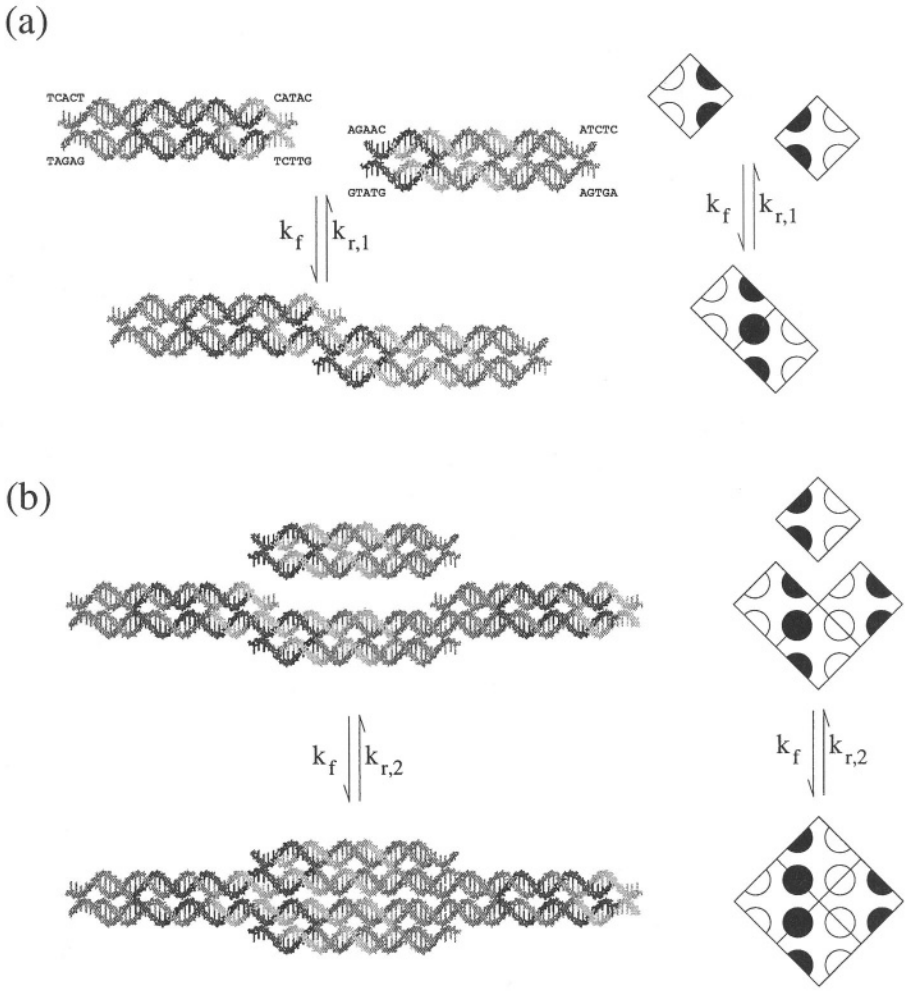


Fig. 2. **(a)** Assembly of two double-crossover tiles via hybridization of 5-nucleotide sticky ends. k_f is the forward rate constant, in */M/sec,* and $k_{r,1} = k_f e^{-G_{se}}$ is the reverse rate constant, in */sec.* **(b)** Assembly of a double-crossover tile into a site on the growth front of a crystal via hybridization of two 5-nucleotide sticky-end pairs. The forward rate constant is assumed to be the same as for the single sticky-end reaction of (a), while the reverse rate constant is assumed to require twice as much energy to simultaneously break both sticky-end bonds – *i.e.,* binding is cooperative – and thus $k_{r,2} = k_f e^{-2G_{se}}$. G_{se} is the free energy of dissociation for a single sticky end, in units of *RT*

fundamental observation is that while on-rates depend only upon the concentration of the tiles, the off-rates depend exponentially upon the total strength of molecular interactions, *i.e.*, the number of base pairs that must be broken in order for the tile to dissociate. Thus, single tiles (*monomers*) that either totally or

partially mismatch their neighbors arrive at a site with equal frequency as tiles that correctly match their neighbors, but the correctly-matching tiles stay much longer. These considerations suggest that behavior of the system is characterized by two essential physical parameters, G_{mc} and G_{se}, respectively measuring the monomer concentration and the sticky-end bond strength as unitless free energies. Specifically, we define $[\text{monomer tile}]/M = e^{-G_{mc}}$, and thus G_{mc} is primarily entropic, as it measures the spatial degrees of freedom that are lost when a free-floating monomer tile is localized on the assembly.[2] Similarly, we define the free energy of dissociation of a single sticky end to be $\Delta G/RT = G_{se}$, and thus G_{se} contains a mix of entropic and enthalpic factors related to the formation of the double-helix, measured in units of *RT*.

We can now formulate the kTAM to describe the growth of a single crystal in an environment where the concentration of monomer tiles remains fixed. Absolute rates for events affecting this crystal are given by

$$r_f = k_f[\text{monomer tile}] = k_f e^{-G_{mc}}$$

for association of a new monomer tile at any given site, and

$$r_{r,b} = k_{r,b} = k_f e^{-bG_{se}}$$

for dissociation of a tile whose interactions with the crystal sum to b, in the "strength units" of the aTAM. b can be thought of as the effective number of unit-strength sticky ends binding the tile to the crystal. These rates specify a continuous-time Markov process (satisfying detailed balance) for modeling the growth of a single crystal in a solution of free monomer tiles.[3]

[2] The simplest situation is when all monomer tile species are present at the same concentration of each species. In some cases, it is convenient to specify the stoichiometry of the tile species, relative to the "generic" monomer tile whose concentration is determined directly by G_{mc}. For example, in the Sierpinski tile simulations, the boundary tiles are present at half the concentration of the rule tiles, which ensures that the boundary grows at approximately the same speed as the interior.

[3] Our model assumes that only single tiles associate to and dissociate from an assembly. Solution will contain a distribution of assembly types, from dimers (two tiles bound to each other) on up. However, near the melting temperature for crystals, where our results hold, dimer concentrations should be significantly lower than monomer concentrations, and thus dimer association events should be rare. Likewise, a pair of connected tiles may simultaneously dissociate from an assembly, or assemblies can even fracture into two or more large pieces. However, the energy required for such events is typically significantly greater than that for monomer dissociation. Therefore, we do not expect our results to change qualitatively if evaluated under more sophisticated models. For the tile sets discussed here, we have seen only minor quantitative changes when either (a) the model also allows tile dissociation events wherein after removal of the tile, the assembly falls apart into two unconnected pieces (such reactions are not reversible in the context of a single-crystal model, and thus are excluded from the standard kTAM), or (b) additionally, the model allows dimers or 2x2 blocks to dissociate together. There are tile sets for which these modification

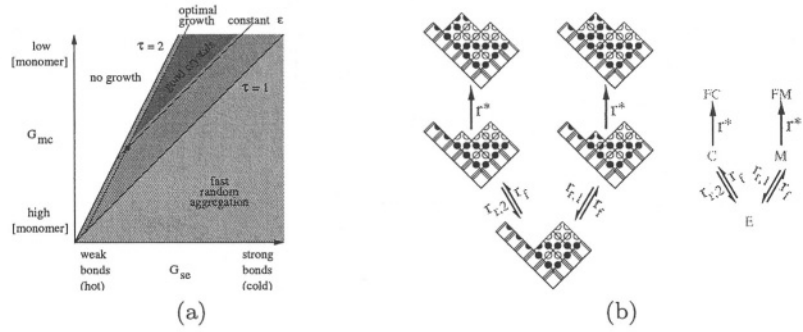


Fig. 3. (a) Phase diagram [28] for crystal growth of tiles implementing a BCA, under the kTAM. "Good crystals" (growth rate comparable to $k_f[DX]$ and error rate smaller than ε) are obtained for large G_{se} and G_{mc}, below the $\tau = 2$ boundary marking the melting transition where $G_{mc} = 2G_{se}$. (b) Model for kinetic trapping. The growth site may *(E)* be empty; *(C)* contain a correct tile; (*M*) contain a mismatched tile; *(FC)* be "frozen" with the correct tile in place; or *(FM)* be "frozen" with the mismatched tile, r^* represents the rate at which tiles on the growth front are covered. The error rate is taken to be the probability that, starting in *E,* the system reaches *FM*

The parameters G_{mc} and G_{se} represent the "physical conditions" under which tile-based assembly can take place. G_{mc} can be made large (or small) by using DNA tiles at low (or high) concentrations. G_{se} can be made large (or small) by letting the self-assembly take place at a cold (or hot) temperature.[4] For what settings of these parameters does the kTAM obey the aTAM rules with high probability? First note that if $2G_{se} > G_{mc} > G_{se}$, then the tile additions shown in figure 1b are favorable, as $r_f > r_{r,2}$, but all other tile additions are unfavorable, as they make at most 1 bond and $r_f < r_{r,1}$. Thus, the aTAM correctly abstracts which reactions are favorable, and which are unfavorable, with respect to the kTAM. However, in the kTAM, unfavorable reactions also occur with some frequency, so we expect assembly errors. Figure 4a shows several snapshots from a Monte Carlo stochastic simulation; single growth errors occur in the 3rd and 4th frames, causing subsequent error-free growth to develop into an undesired pattern. How frequent are these errors, and how can they be minimized?

do have qualitatively significant ramifications, for example, tile sets involving linear polymerization or blocks of tiles that are strongly bound to each other but have weak interactions with the crystal.

[4] Naturally, the assumption that G_{mc} and G_{se} both remain constant is likely to be violated in actual experiments, both for reasons under our control (*e.g.,* using a temperature annealing schedule) and for reasons not easily under our control (*e.g.,* the depletion of ambient monomer tile concentrations as a significant fraction of tiles become incorporated into crystal assemblies.

Previous studies [28] using this model showed that both growth errors and certain nucleation errors[5] can be controlled in the limit of low monomer tile concentrations (*i.e.,* large G_{mc}) and strong sticky-end interactions (*i.e.,* large G_{se}) so long as the system is kept near the melting temperature of the crystal (*i.e.,* $G_{mc} \approx 2G_{se}$). As shown in figure 3a, phase space can be divided into three regions: no crystal growth occurs if $G_{mc} > 2G_{se}$; algorithmic self-assembly (with some error rate) occurs for $2G_{se} > G_{mc} > G_{se}$; and essentially random aggregation is obtained for $G_{se} > G_{mc}$. Within the algorithmic phase, two factors limit the performance that can be achieved: thermodynamics tells us the best error rate that can be achieved at equilibrium given the energetics of the system, while the kinetics determine how quickly we get there – if at all.

If self-assembly achieves equilibrium, the probability of observing a particular assembly A will be governed by the Boltzman equation:

$$Pr(A) = \frac{1}{Z} e^{-G(A)} \quad \text{with} \quad Z = \sum_{A'} e^{-G(A')}$$

where $G(A) = nG_{mc} - bG_{se}$ is the free energy of the assembly, n is the number of tiles in the assembly, b is the total of all bond strengths in the assembly, and Z is the partition function. Thus, an n-tile assembly that has Δb more bond strength than another n-tile assembly will be $e^{\Delta b G_{se}}$ more likely. In a direct BCA tile set with $(N + 1)$ states, a typical growth site will present two sides with strength-1 bonds, a unique correct tile will match both bonds, and there will be exactly N competing tiles that have a mismatch on the left side, as well as N that have a mismatch on the right side. It follows that the equilibrium probability of incorporating the correct tile at a given growth site is

$$1 - \varepsilon \approx \frac{e^{2G_{se}}}{e^{2G_{se}} + 2Ne^{G_{se}}}, \text{ and thus } \varepsilon \approx 2Ne^{-G_{se}},$$

where ε is the per-tile error rate. It is mildly surprising that G_{mc} has no effect on the equilibrium error rates.

However, due to kinetics, equilibrium is seldom achieved far below the melting transition. The primary cause of growth errors in this case was found to be a form of kinetic trapping, wherein tiles that associate with a mismatch on the growth front don't have time to dissociate, and are frozen in place by further growth; thus equilibrium error rates are observed only near the melting transition. The essential feature of kinetic trapping within BCA tile self-assembly is that once an error has occurred, both sites above the mismatched tile display an (x, y) pair that is perfectly matched by some monomer tile in solution, because tiles implementing all replacement rules $(x, y) \rightarrow (f_1(x, y), f_2(x, y))$ are present. Thus,

[5] It was shown that spurious nucleation of rule tiles can be controlled, for essentially the same reason that supersaturated solutions can be maintained: there is a critical nucleus size (based on surface-to-volume energies) beyond which growth is favorable and below which growth is unfavorable. However, spurious nucleation of boundaries is a more difficult issue [21].

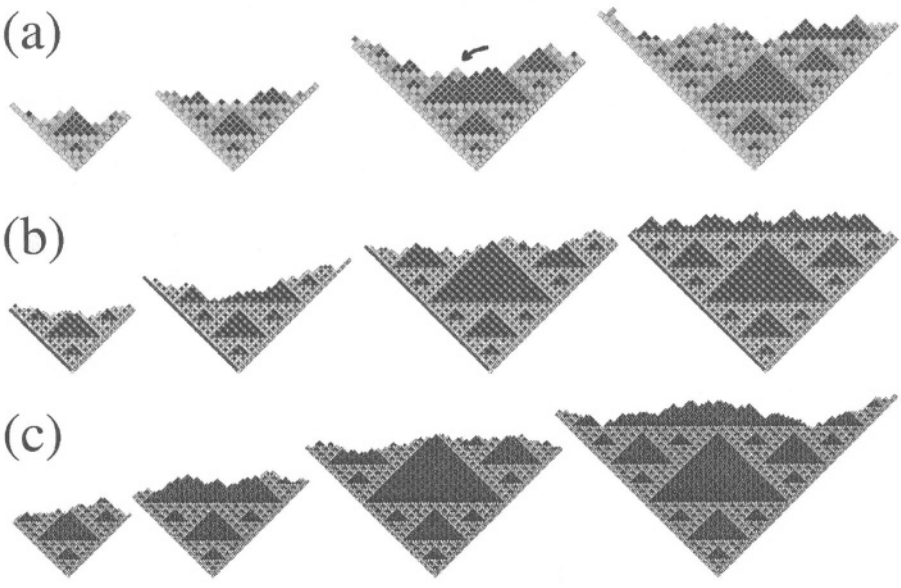


Fig. 4. **(a)** Growth of the original (1 × 1) Sierpinski tile set at $G_{mc} = 13.9$ and $G_{se} = 7.0$, to a size of ~ 32 layers in ~ 530 simulated seconds. Two errors can be seen; the first occurs in the third frame and is indicated by an arrow. Subsequent error-free growth correctly propagates the erroneous information, **(b)** Growth of the 2 × 2 proofreading tiles at $G_{mc} = 12.9$ and $G_{se} = 6.5$, to a size of ~ 64 layers in ~ 460 simulated seconds. **(c)** Growth of the 3 × 3 proofreading tiles at $G_{mc} = 11.9$ and $G_{se} = 6.0$, to a size of ~ 96 layers in ~ 310 simulated seconds

if such a tile arrives before the mismatched tile dissociates, the mismatched tile becomes locked in by multiple bonds, and is now unlikely to dissociate. For a direct BCA tile set, the kinetic trap model shown in figure 3b accurately predicts growth errors in kTAM simulations to obey [28]

$$1 - \varepsilon \approx \frac{1}{1 + 2N\frac{r^* + r_{r,2}}{r^* + r_{r,1}}} \text{ and thus } \varepsilon \approx 2N\frac{r^* + r_{r,2}}{r^* + r_{r,1}} \underset{r^* \to 0}{\longrightarrow} 2Ne^{-G_{se}},$$

where $r = \frac{1}{2}(r_f - r_{r,2})$ is the overall growth rate and $r^* = \alpha r$ is the effective rate at which sites are frozen in the model. $\alpha = 1.5$ is a free parameter chosen fit to the data; in some sense, it accounts for the fluctuations of the growth process, in which the growth front will wash back and forth over a given site several times before it is "frozen" in place.

The kinetic trapping theory identified a critical relationship. Although arbitrarily low error rates can be achieved by appropriate choice of G_{mc} and G_{se}, they come at the cost of a significant slow-down. This trade-off can be visualized by plotting r *vs* ε for all reasonable values of G_{mc} and G_{se}. As illustrated by the upper (1 × 1) plots in figure 5, all points lie above $r \approx \beta\varepsilon^2$, where $\beta \approx 1 \times 10^4 / M/sec$ is determined by how far below the melting temperature gives

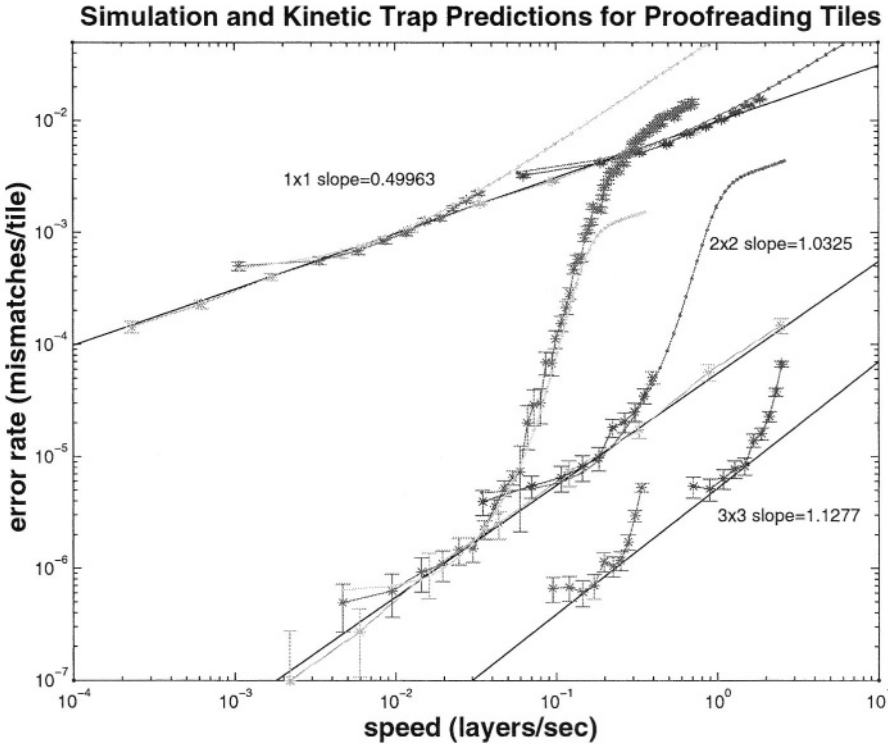


Fig. 5. Plot of error rate *vs.* growth speed as measured in simulations (*'s) and as according to kinetic trap theory (lines). Here, $G_{mc} = 2{*}G_{se} - \epsilon$; *i.e.,* ϵ measures how far the reaction is from the $\tau = 2$ melting transition. Curved lines (red and blue) vary G_{mc} and ϵ for fixed G_{se}, thus demonstrating how the growth speed decreases too near the melting transition and the error rates increases too far below the melting transition. (Simulations used $G_{se} = 6.5$ and 8.5 for 1×1, 6.5 and 7.5 for 2×2, and 5.5 and 6.5 for 3×3 tile sets.) Straight lines (green) vary G_{se} and G_{mc} for fixed ϵ, thus following the line just below the melting transition in phase space to obtain a Pareto-optimal speed/error trade-off. (Simulations used $k_f = 10^6/M/sec$ and $\epsilon = .2, .1, .15$ for 1×1, 2×2, and 3×3 tile sets respectively. Error bars show two standard deviations, computed using $\sigma = \sqrt{\varepsilon}/\sqrt{m}$ where m is the total number of tiles grown in all simulations using a given G_{se} and G_{mc}. A variable number of simulation runs were used, chosen to make the error bars small.)

the optimal trade-off.[6] This defines the Pareto-optimal boundary along which r is the fastest growth rate that achieves error rate ε. Thus, decreasing error rates by a factor of 10 entails slowing down the self-assembly process by a factor of 100. This is bad; results on fault-tolerant computing in the digital circuit model typically entail a logarithmic, rather than quadratic, slow-down [24, 15].

[6] If $G_{mc} = 2G_{se} - \epsilon$, then $r = \frac{1}{2}(r_f - r_{r,2}) = \frac{1}{2}k_f(e^{\epsilon} - 1)e^{-2G_{se}} \approx \frac{k_f}{8N^2}\varepsilon^2$ for equilibrium error rates. This estimate is accurate only within an order of magnitude.

In the rest of the paper, we give a construction of "proofreading" tile sets to implement arbitrary BCA, in which each original tile is replaced by a $K \times K$ block of tiles in the new tile set.[7] Simulation and theory results for 2×2 and 3×3 tile sets are shown in figures 4 and 5, showing scaling behavior of roughly $r \approx \beta\varepsilon^{1.0}$ and $r \approx \beta\varepsilon^{0.89}$ respectively, with β varying by less than a factor of five depending on the tile set. This is a significant improvement: for target error rates of 10^{-4}, the 2×2 proofreading tiles result in a 10^4-fold speed-up over the original tile set. Similarly, for the same physical conditions that result in a 1% error rate for the original tile set, the 2×2 proofreading tile set yields a 0.01% error rate, and the 3×3 proofreading tile set yields a 0.001% error rate. This is extremely encouraging for experimental studies of algorithmic self-assembly: conditions in which perfect 10×10 Sierpinski triangles can now be grown may yield 100×100 triangles with proofreading tiles. Still, these constructions result in greater slow-down than achieved in the digital circuit model, suggesting that further improvements await discovery.

The remainder of this paper describes the proofreading tile set construction, gives an explanation of the principles by which it works, and comments on its limitations.

3 Proofreading Tile Sets

The ability for the kTAM to discriminate between tiles that partially or perfectly match a growth site relies on the cooperativity of sticky-end binding: the binding of one sticky end stabilizes the binding of the other sticky end. The basic idea of proofreading tiles is to exploit cooperative binding at the next higher level: to have several tiles that stabilize each other when they bind together. Cooperative binding is a common feature of transcription factors in genetic regulatory networks [16] and has been examined as a mechanism for increased sensitivity in one-dimensional self-assembly processes [2, 3]. New issues arise in the context of two-dimensional self-assembly.

The general 2×2 proofreading construction is shown in figure 6a, and its application to the Sierpinski tile set is shown in figure 6bcd. Essentially, each rule tile in the original tile set is replaced by four tiles with related labels. Arranged in a 2×2 block, the sides of the block present the same logical labels as the original tile. The side internal to the block are given unique labels, not shared by tiles from any other block. Thus, assembly from the seed tile according to the aTAM proceeds according to the same logic as the original tile set, but scaled up in size by a factor of two.[8] However, in the kTAM, a new phenomenon can be observed when a mismatched tile is incorporated: there is now *no way to continue growth without making an additional error.* This is illustrated by the small tiles in figure 6d: after the initial (lowest) small tile arrives, forming a mismatch

[7] The direct BCA tile sets can be considered to be the 1×1 proofreading construction.

[8] For self-similar patterns like the Sierpinski triangle, the resolution of the resulting pattern remains the same – each 2×2 block can be labeled according to the pre-computed pattern.

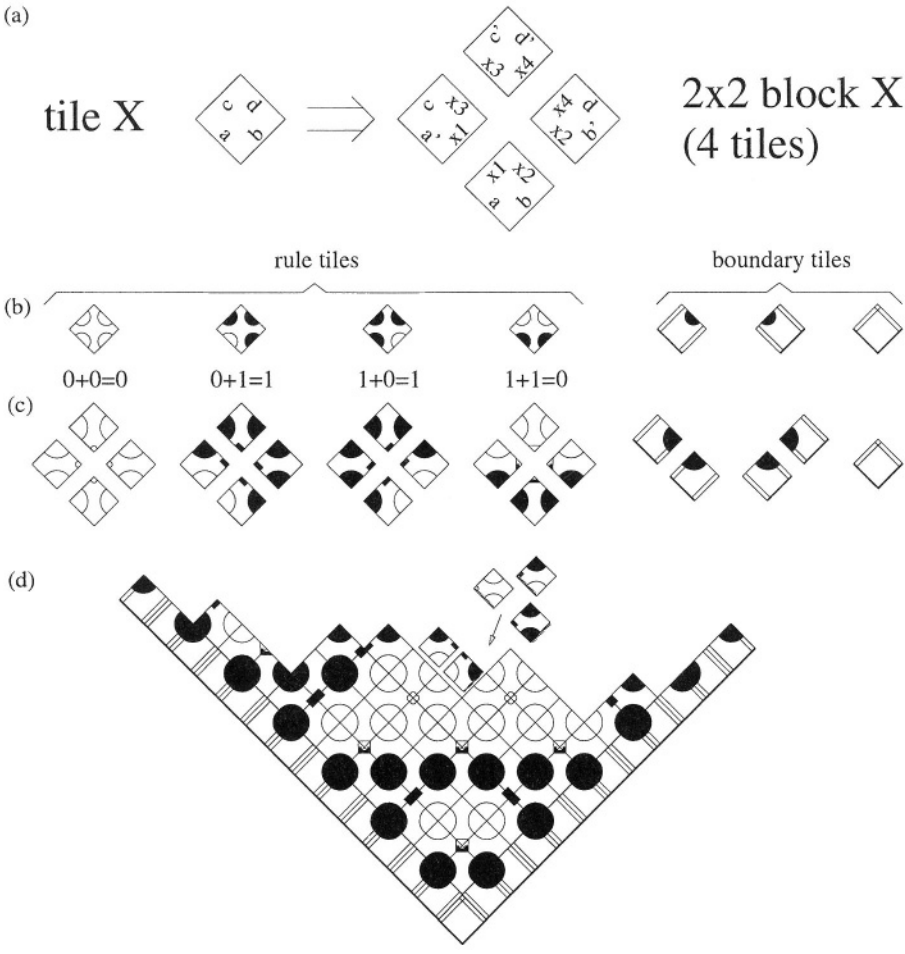


Fig. 6. **(a)** The general 2×2 proofreading construction for rule tiles, **(b)** The original Sierpinski tiles, **(c)** The 2×2 proofreading Sierpinski tiles. **(d)** Growth of the proofreading Sierpinski tiles. Small tiles illustrate that when a mismatched tile is incorporated, further growth on one side must involve a second mismatch

on one side, any further tile assembling on that side will either (a) agree with the initial tile but, because it therefore must be part of the same proofreading block, mismatch on its lower right side, or (b) agree with its lower right input, but therefore form a mismatch with the initial small tile. The assembly process stalls, giving time for the initial mismatched tile to fall off and be replaced by a correct tile. The final assembly therefore has no record of the mishap having occurred.

The forcing of errors to be co-localized in pairs results in the error rate being squared relative to the original tile set. A detailed kinetic trapping model for

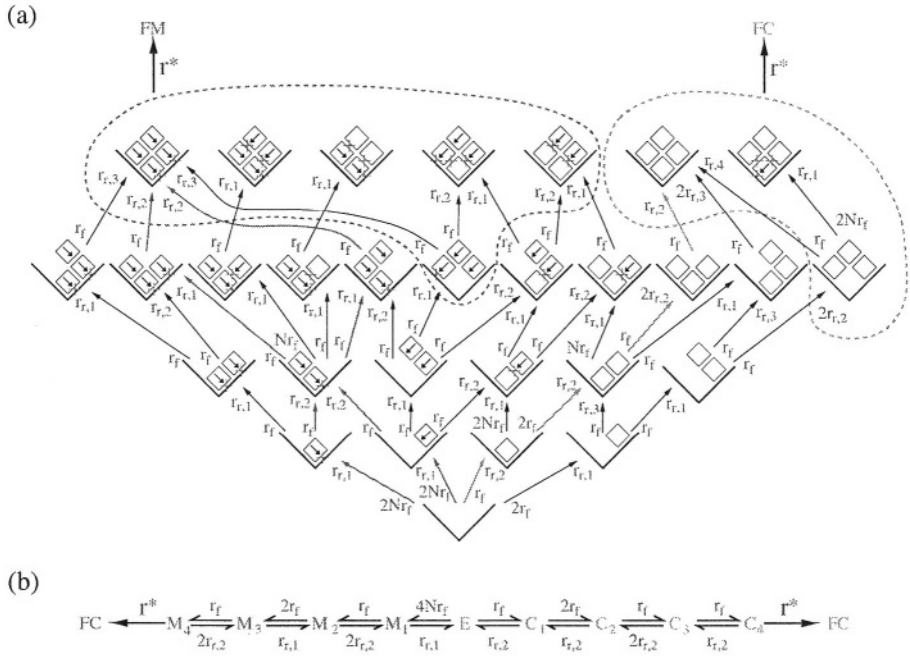


Fig. 7. **(a)** A kinetic trapping model including all 29 states (up to symmetry) representing "well-associated" tiles within a 2 × 2 growth site (see text for details). Arrows inside tiles indicate that the tile belongs to a proofreading block that has a mismatch to the input in the indicated direction; there are N such tiles for an $(N + 1)$-state BCA. Arrows between states indicate reversible reactions (association or dissociation of a tile); reverse reaction rates are given at the head of each arrow, and forward reaction rates are given near the tail. States within dotted circles each have an irreversible reaction to a frozen state (either FM or FC) with rate r^*. **(b)** A simplified kinetic trapping model with 9 states considers only the major reaction pathways in (a), which are indicated by the red and green reaction arrows for pathways leading to mismatched or correct blocks, respectively

2 × 2 proofreading tiles (shown in figure 7a) produces excellent agreement[9] with the simulation results (shown in figure 5), including the precipitous decline in reliability as physical conditions move away from the melting transition. Each state in this model represents a possible arrangement of tiles (up to symmetries) within the 2 × 2 growth site. Each tile could be either part of the correct block for that site (unlabeled) or part of a block that has a mismatch on one side or the other (indicated by the direction of the arrow; there are N such blocks for BCA tile sets). The 29 states considered are all those in which the tiles are "well-associated", that is, we prune the full model by removing all states in

[9] Again, the value of r^* was determined by fitting free parameters to best match the data. Here, we used $r^* = \alpha r e^{\gamma\epsilon}$ where $\epsilon = 2G_{se} - G_{mc}$, $\alpha = 1$, and $\gamma = 12$. We have no strong justification for this formula.

which more than one tile is attached by only one bond or in which some tile is attached by no bonds at all – such states would be very short-lived.

A simplified model (shown in figure 7b) considers just the dominant pathways (red and green reaction arrows in figure 7a), and yields similar results near the melting transition. The improvement in the error rate near the melting transition, then, is due to the elementary features preserved in this model, namely that (a) the correct block can be formed via a series of four favorable steps, and (b) every path to a mismatch contains at least two significantly unfavorable steps with a fast $r_{r,1}$ reverse reaction.

Although baroque, both models can easily be solved numerically by representing the transition rates in matrix form and computing the steady-state by matrix inverse.

The basic phenomenon can be more easily understood using the following intuition. Consider first the direct BCA tile set. Optimal growth rates are obtained near the melting temperature of the crystals, where $G_{mc} \approx 2G_{se}$. Under these conditions, the error rate reaches the thermodynamic limit of $\varepsilon \approx e^{-\Delta G/RT} = e^{-G_{se}}$, where ΔG is the difference in free energy between an assembly with a mismatched tile and one with a correct tile. However, the growth rate will be proportional to the monomer tile concentration, $r \approx \beta[DX] = \beta e^{G_{mc}}$. This yields the scaling relation for the original tile set, $r \approx \beta\varepsilon^2$.

This type of argument can be generalized for $K \times K$ proofreading constructions, wherein each tile is replaced by a $K \times K$ block of unique tiles. Optimal growth rates still occur near the melting temperature ($G_{mc} \approx 2G_{se}$), and the growth rate is still $r \approx \beta[DX] = \beta e^{G_{mc}}$. However, the thermodynamic error rate (for an entire block) is now determined by the minimal error, which involves at least K mismatched tiles.[10] Thus[11] $\varepsilon \approx e^{-\Delta G/RT} = e^{-KG_{se}}$, and the argument yields $r \approx \beta\varepsilon^{2/K}$.

If the above reasoning held for all K, and the proofreading block size were to be chosen based on the target error rate needed, a logarithmic dependence of K on the target error rate would be achieved for constant physical conditions. This matches what has been found for digital circuits. Unfortunately, although the argument correctly predicts the scaling behavior for $K = 2$, the simulation results for $K = 3$ (figure 5) and $K = 4$ (data not shown) fall short of the prediction.

We attribute this failure to a second mechanism for growth errors. Whereas proofreading tiles perform well at correcting errors during growth at a site where correct growth *could* occur, they do nothing to prevent errors due to spontaneous

[10] Since each erroneous block involves K^2 tiles and typically contains exactly K mismatches, the per-block error rates $\varepsilon_{block} = K\varepsilon$, where ε is the per-tile error fraction. (Tile errors are now clearly not iid.) We found it more convenient to report simulation results in terms of the per-tile error rate.

[11] This argument illustrates why it is necessary to use cooperativity by relying on the independent assembly of the K^2 pieces in a block. If one were to use a preassembled block, the melting temperature would occur at $G_{mc} \approx 2KG_{se}$, giving rise to the original $r \approx \beta\varepsilon^2$ scaling behavior.

growth on a facet ("roughening"), as shown in figure 10a. Ensuring that boundaries grow at approximately the same speed as the interior[12] reduces the amount of faceting, but even so, facets of length n appear with a frequency of $\approx 2^{-n}$, making facet growth errors rare but unavoidable. Some other error-correcting strategy will be necessary to prevent this type of error.

4 Strong Bonds, Capping Tiles, and Self-Repair

It is important to realize that the proofreading tile set construction given here doesn't work for all tile sets. The existence of the alternative growth error mechanism on facets implies that tile sets whose growth process intrinsically involves facets will fail to derive great benefit from proofreading tiles. For example, the the tile set presented in [20] for growing $M \times M$ squares using $O(\log M)$ tiles results in a final assembly in which three sides are facets – *i.e.,* each tile on those sides displays a strength-1 bond, but under $\tau = 2$ aTAM rules, no growth can occur. In the kTAM, growth will eventually occur, ruining the desired structure, as shown in figure 10b(top) for a 26×26 square. Proofreading tiles do nothing to fix this problem.

To reliably construct an $M \times M$ square required modifying the original tile set in three ways: first, identifying the direction of growth and specializing the tiles so that each tile is used for only one direction of growth, and each sticky end appears on as few tiles as possible; second, making use of additional "capping" tiles to quickly cover the final facets of the square with tiles that have null bonds on their outer sides, thus reducing errors due to roughening; and third, using two perpendicular binary counters to encourage the growth front to avoid large facets. Assembly using this improved tile set is shown in figure 10b(middle) for a 49×49 square.

With facet roughening errors ameliorated, it makes sense to ask whether now proofreading further decreases the error rate. However, because growth of the square within the counter region takes a meandering path making use of several rule tiles with strength-2 bonds, the proofreading construction must be extended to blocks involving strength-2 bonds. This construction, shown in figure 10c, can only be used for tile sets in which each tile occurs only during growth in a particular direction; *i.e.,* its input and output sides can be determined and are used consistently. The bond labels on proofreading tiles are as described previously, but the strength of those bonds is determined by the growth direction (for the input sides) or by the growth direction of subsequent tiles (for the output sides). This 2×2 proofreading construction was applied to the improved tile set described above, with results shown in figure 10b(bottom).

Out of 31 trials, the original tile set never formed the desired 26×26 squares properly, the improved tile set was 65% successful at forming 49×49 squares, and the proofreading tile set formed 98×98 squares 87% of the time. For comparison, the improved tile set formed 99×99 squares only 26% of the time. The

[12] This is the case in the simulations, thanks to the boundary tile concentration being set to $\frac{1}{2}$ the concentration of the rule tiles.

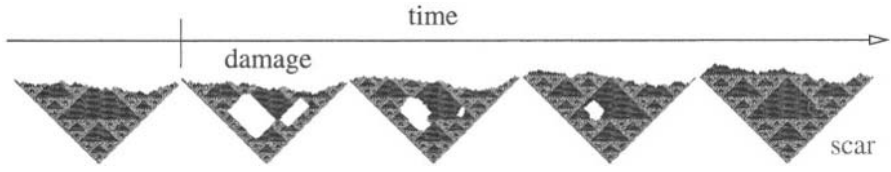


Fig. 8. Proofreading tile sets are often able to heal a puncture in the crystal. Sometimes, as in this case, some of the tiles that fill in the puncture do not perfectly match their neighbors – a form of "scar tissue"

disappointingly modest[13] benefit provided by the proofreading tiles suggests that alternative error mechanisms are dominant for these tile sets.

The proofreading tiles also give rise to some surprising, and pleasant, behaviors. One such phenomenon is their ability to heal punctures of the growing crystal, as shown in figure 8. Since the identity of tiles within the punctured region is uniquely determined by the perimeter of that region (in fact, just the lower portion of the perimeter suffices), one might expect even the non-proofreading tile set to be able to correctly fill in the punctured region. However, regrowth into the punctured region may occur in any direction, including backward from the most advanced edge of the puncture, where there are multiple ways to proceed locally when using tiles implementing irreversible BCA, such as the Sierpinski tiles.[14] Thus, regrowth is not always perfect; indeed, the direct BCA tiles very often leave "scar tissue", although the proofreading tiles do so much less frequently.

5 Discussion

The primary result of this paper is that dramatic improvements in the error rates for algorithmic self-assembly can be achieved, in principle, by proper redesign of the abstract tile set, without changing the fundamental molecular implementation. The 2×2 proofreading tiles for BCA increase the optimal growth speed r for which a target error rate ϵ can be achieved from $r = \beta\epsilon^2$ to $r = \beta\epsilon$, and greater improvements can be obtained with larger $K \times K$ proofreading constructions. If these constructions hold up in practice, it may be possible for algorithmic self-assembly to scale up to macroscopic assemblies without errors.

Although the theoretical and simulation results presented here firmly establish the effectiveness of the proofreading tile construction, the rigorous proof of their robustness in the kTAM remains an open problem.

When considering implementation of proofreading tile sets with DNA tiles, the question of efficiency arises. One wishes for a minimal number of tiles, since each additional tile requires a significant amount of laboratory work. Fortunately, for tile sets implementing BCA for which both outputs are identical

[13] The 2×2 proofreading Sierpinski tile set under these conditions would obtain $\varepsilon = 1.5 \times 10^{-7}$, and therefore 10^4 tiles would assemble without errors 99.9% of the time.

[14] As pointed out by Adleman (personal communication), this suggests a relationship between reversible cellular automaton logic and self-healing properties.

(*i.e.* $f_1 = f_2 = f$, as is the case for the Sierpinski tiles), blocks that output the same value $f(x, y)$ can share the same $(K - 1) \times (K - 1)$ sub-block, as shown in figure 9, achieving approximately the same fault-tolerance while using only $(2K - 1)N^2 + (K - 1)^2 N$ instead of $K^2 N^2$ rule tiles. For DNA tiles made from DAO-E molecules, for which two DNA tile must be created for each abstract tile (subject to exploitation of symmetries) [21], 2×2 proofreading increases the number of DNA rule tiles from 6 for the original Sierpinski tile set to 9 for proofreading – a very moderate increase. We are therefore exploring this tile set experimentally.

Other optimizations are possible, such as trying to reduce the increase in scale of the final pattern produced by self-assembly. Reif [17] has already proposed constructions that improve upon the K-fold spatial scaling inherent here.

Furthermore, it is at present unclear how these constructions can be adapted for general tile sets, where the growth front may involve significant faceting. It appears that an additional proofreading mechanism will be required to correct for errors that occur due to premature growth on a facet; the concept of "invadable tiles" may help here [6].

Robustness to assembly errors may be contrasted with robustness to implementation errors, wherein the molecular tiles fail to conform to the desired specifications, as expressed within the kTAM. For example, bond strengths for different labels that should all be identical (*e.g.,* weak) may vary somewhat in a specific set of DNA tiles. Similarly, an experiment may provide the tiles at slightly different (rather than identical) concentrations. More seriously, as assembly proceeds in an undisturbed reaction, monomer tile concentrations will be depleted over time. Also, monomer tiles may aggregate into small assemblies, rather than accreting one-by-one onto the growing crystal. It is hoped that proofreading tiles, in addition to providing robustness to assembly errors, will provide improved robustness to implementation errors, but this issue has not yet been investigated.

Finally, our experience with experimental systems suggests that a major remaining issue is constraining assembly to begin only from the selected seed tile, *i.e.,* minimizing spontaneous spurious nucleation of rule tiles or boundary tiles. Careful exploitation of critical nucleus size during supersaturation appears to provide a solution to this quandary [22].

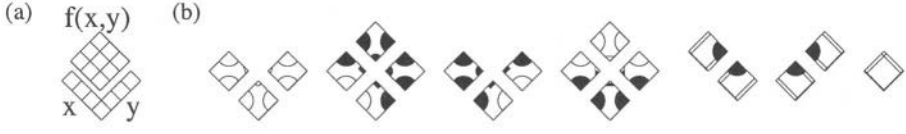


Fig. 9. **(a)** Schema for the optimized tile set, in which every pair of inputs x, y is given unique tiles for the edge of a $K \times K$ block, and the remaining $(K - 1) \times (K - 1)$ tiles is shared for each output $f(x, y)$. **(b)** An optimized 2×2 proofreading tile set for the Sierpinski tiles

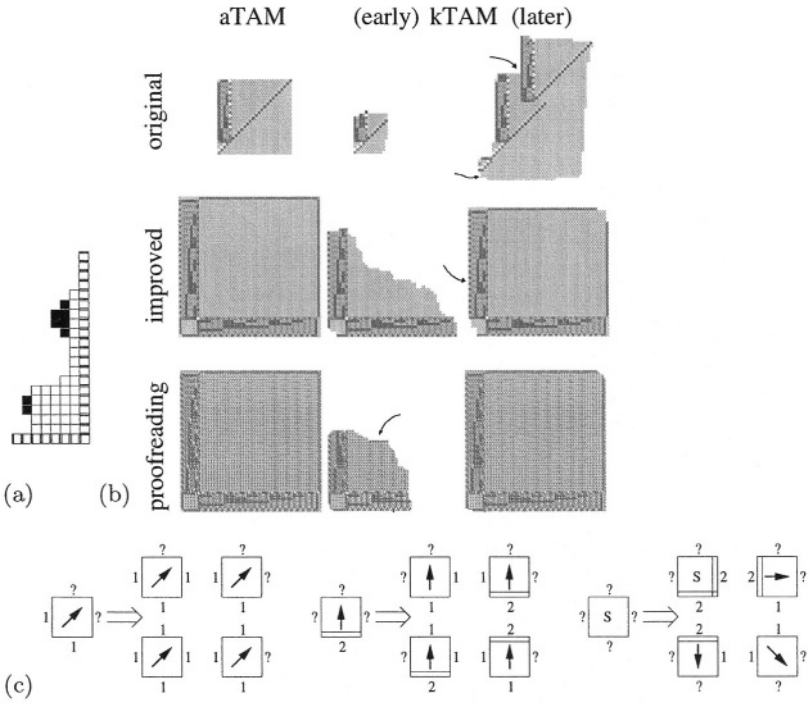


Fig. 10. **(a)** Small black tiles illustrate spurious nucleation of the subsequent layer (facet roughening). The information content of these tiles is likely to be wrong. Our proofreading tile set construction does not efficiently correct these errors, **(b)** From left to right: Assembly according to the aTAM at $\tau = 2$; early growth in the kTAM at $G_{mc} = 17.7$ and $G_{se} = 9.0$; and the resulting structure. From top to bottom: The binary counter construction for assembling $M \times M$ squares; the improved two-counter construction with capping tiles; and the 2×2 proofreading construction for the latter. All tile sets assemble perfectly under the aTAM. Under the kTAM, the one-counter tile set fails even for a small square because spurious nucleation on facets frequently results in uncontrolled growth (arrows). The two-counter tile set still suffers from algorithmic errors (arrow), but the capping tiles significantly reduce spurious nucleation of new layers. The proofreading construction reduces these problems only somewhat for this tile set. The arrow indicates a facet roughening error in which capping tiles prematurely nucleated on the growth front. In this run, these capping tiles are eventually displaced by the growth of correct tiles, **(c)** The scheme for assigning bond strengths for proofreading tiles when the tile set contains strength-2 bonds. Arrows indicate the direction of growth for correct use of the tile, and "S" indicates the seed tile. The strength of bonds on sides marked by "?" is dictated by the tiles that bind to them in a correct assembly; if no tiles bind to them, strength-0 bonds are used

Acknowledgements

This work benefited from discussions with Leonard Adleman, Matthew Cook, Ashish Goel, Paul Rothemund, Rebecca Schulman, Georg Seelig, David Soloveichik, and Chris Umans. Thanks to John Reif for encouraging me to write this up and for sharing his unpublished manuscript. EW and RB were supported by NSF CAREER Grant No. 0093486, DARPA Bio-Computation Contract F30602-01-2-0561, NASA NRA2-37143, and GenTel. Simulation code and tile sets used in this paper, as well as MATLAB scripts for evaluating the kinetic trapping models, may be obtained from `http://www.dna.caltech.edu/SupplementaryMaterial`.

References

[1] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science,* 266:1021–1024, November 11, 1994.

[2] Roy Bar-Ziv and Albert Libchaber. Effects of DNA sequence and structure on binding of RecA to single-stranded DNA. *Proc. Nat. Acad. Sci. USA,* 98(16):9068–9073, 2001.

[3] Roy Bar-Ziv, Tsvi Tlusty, and Albert Libchaber. Protein-DNA computation by stochastic assembly cascade. *Proc. Nat. Acad. Sci. USA,* 99(18): 11589–11592, 2002.

[4] Victor A. Bloomfield, Donald M. Crothers, and Ignacio Tinoco, Jr. *Nucleic Acids: Structures, Properties, and Functions.* University Science Books, 2000.

[5] Boris A. Bondarenko. *Generalized Pascal Triangles and Pyramids, Their Fractals, Graphs and Applications.* The Fibonacci Association, 1993. Translated from the Russion and edited by Richard C. Bollinger.

[6] Ho-Lin Chen, Qi Cheng, Ashish Goel, Ming deh Huang, and Pablo Moisset de Espanés. Invadable self-assembly: Combining robustness with efficiency. *ACM-SIAM Symposium on Discrete Algorithms (SODA),* to appear, 2004.

[7] Junghuei Chen and John Reif, editors. *DNA Computing 9,* Berlin Heidelberg, to appear. Springer-Verlag.

[8] Matthew Cook, Paul Wilhelm Karl Rothemund, and Erik Winfree. Self-assembled circuit patterns. In Chen and Reif [7].

[9] Tsu-Ju Fu and Nadrian C. Seeman. DNA double-crossover molecules. *Biochemistry,* 32:3211–3220, 1993.

[10] Peter Gács. Reliable cellular automata with self-organization. *Journal of Statistical Physics,* 103(l/2):45–267, 2001.

[11] Thomas H. LaBean, Hao Yan, Jens Kopatsch, Furong Liu, Erik Winfree, John H. Reif, and Nadrian C. Seeman. Construction, analysis, ligation, and self-assembly of DNA triple crossover complexes. *Journal of the American Chemical Society,* 122:1848–1860, 2000.

[12] Michail G. Lagoudakis and Thomas H. LaBean. 2-D DNA self-assembly for satisfiability. In Erik Winfree and David K. Gifford, editors, *DNA Based Computers V,* volume 54 of *DIMACS,* pages 141–154, Providence, RI, 2000. American Mathematical Society.

[13] Chengde Mao, Thomas H. LaBean, John H. Reif, and Nadrian C. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature,* 407(6803):493–496, 2000.

[14] Chengde Mao, Weiqiong Sun, and Nadrian C. Seeman. Designed two-dimensional DNA Holliday junction arrays visualized by atomic force microscopy. *Journal of the American Chemical Society,* 121(23):5437–5443, 1999.

[15] Nickolas Pippenger. Developments in "the synthesis of reliable organisms from unreliable components". In *The Legacy of John von Neumann,* pages 311–324. American Mathematical Society, 1990.

[16] Mark Ptashne. *A Genetic Switch, 2nd ed.* Cell Press & Blackwell, 1992.

[17] John Reif. Compact error-resilient computational DNA tiling assemblies. Unpublished manuscript.

[18] John Reif. Local parallel biomolecular computing. In Harvey Rubin and David Harlan Wood, editors, *DNA Based Computers III,* volume 48 of *DIMACS,* pages 217–254, Providence, RI, 1999. American Mathematical Society.

[19] Paul W. K. Rothemund and Erik Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. In preparation.

[20] Paul Wilhelm Karl Rothemund and Erik Winfree. The program-size complexity of self-assembled squares. In *Symposium on Theory of Computing (STOC).* ACM, 2000.

[21] Rebecca Schulman, Shaun Lee, Nick Papadakis, and Erik Winfree. One dimensional boundaries for DNA tile assembly. In Chen and Reif [7].

[22] Rebecca Schulman and Erik Winfree. Controlling nucleation rates in algorithmic self-assembly. In preparation.

[23] David Soloveichik and Erik Winfree. Complexity of self-assembled scale-invariant shapes. Submitted.

[24] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies,* pages 43–98. Princeton University Press, 1956.

[25] Hao Wang. Proving theorems by pattern recognition. II. *Bell System Technical Journal,* 40:1–42, 1961.

[26] Hao Wang. Dominoes and the AEA case of the decision problem. In Jerome Fox, editor, *Proceedings of the Symposium on the Mathematical Theory of Automata,* pages 23–55, Brooklyn, New York, 1963. Polytechnic Press.

[27] Erik Winfree. On the computational power of DNA annealing and ligation. In Richard J. Lipton and Eric B. Baum, editors, *DNA Based Computers,* volume 27 of *DIMACS,* pages 199–221, Providence, RI, 1996. American Mathematical Society.

[28] Erik Winfree. Simulations of computing by self-assembly. Technical Report CS-TR:1998.22, Caltech, 1998.

[29] Erik Winfree, Furong Liu, Lisa A. Wenzler, and Nadrian C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature,* 394:539–544, 1998.

[30] Erik Winfree, Xiaoping Yang, and Nadrian C. Seeman. Universal computation via self-assembly of DNA: Some theory and experiments. In Laura F. Landweber and Eric B. Baum, editors, *DNA Based Computers II,* volume 44 of *DIMACS,* pages 191–213, Providence, RI, 1998. American Mathematical Society.

[31] Hao Yan, Sung Ha Park, Gleb Finkelstein, John H. Reif, and Thomas H. LaBean. DNA-templated self-assembly of protein arrays and highly conductive nanowires. *Science,* 301:1882–1884, 2003.

A DNA-Based Memory with *In Vitro* Learning and Associative Recall

Junghuei Chen[1], Russell Deaton[2], and Yu-Zhen Wang[1]

[1] Chemistry and Biochemistry, The University of Delaware
Newark, DE, 19716
junghuei@udel.edu
[2] Computer Science and Engineering, The University of Arkansas
Fayetteville, AR 72701
rdeaton@uark.edu

Abstract. A DNA-based memory was implemented with *in vitro* learning and associative recall. The learning protocol stores the sequences to which it is exposed, and memories are recalled by sequence content through DNA-to-DNA template annealing reactions. Experiments demonstrated that biological DNA could be learned, that sequences similar to the training DNA were recalled correctly, and that unlike sequences were differentiated. Theoretical estimates indicate that the memory has a pattern separation capability that is very large, and that it can learn long DNA sequences. The learning and recall protocols are massively parallel, as well as simple, inexpensive, and quick. The memory has several potential applications in detection and classification of biological sequences, as well as a massive storage capacity for non-biological data.

1 Introduction

In DNA Computing (DNAC), the information capacity of biomolecules and the tools of molecular biology are exploited for non-biological or computational purposes. The properties of DNA that are most useful for this are its ability to store information in the sequence of nucleotide bases and to have that information manipulated by various enzymes and laboratory procedures. Advantages for computing with DNA are its ability to store vast amounts of information in small volumes and the massive parallelism with which DNA can be manipulated and searched. The processing of the information stored in DNA, however, is neither exact nor deterministic, but rather random, incomplete, and complex, especially as the size and sequence diversity of the oligonucleotide mix increases. For example, enzymatic reactions can go to varying degrees of completion, and matching of the information stored in DNA through template-matching hybridization reactions is inexact.

Adleman[l] was the first to demonstrate the inherent computational capability of DNA by solving an instance of the Hamiltonian Path Problem (HPP). Later, Lipton[2] showed how to generalize Adleman's approach to solve satisfiability problems (SAT). The current state of the art is a recent solution of

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 145–156, 2004.

a 20 variable SAT problem[3]. In Adleman-Lipton DNA computation, programs consist of DNA sequences designed so that hybridizations search for and form the solution, and laboratory procedures to extract the solution. To avoid errors, DNA sequences are designed to hybridize as planned, which is a nontrivial problem[4, 5, 6, 7]. In addition, there must be enough DNA in the test tube to represent all possible solutions, which leads to scaling problems[8].

As recognized by Baum[9], the imprecision in the DNA-to-DNA template-matching hybridization reaction could be used to advantage to implement a content-addressable, or associative, recall mechanism for a memory with a theoretical capacity larger than the human brain. In such a memory, retrieval of information is not accomplished by looking up an address, but by recalling those memories whose content is associated with or close to the given stimulus. This could be implemented in DNA by storing information in DNA oligonucleotide sequences. Inputs are also represented by DNA oligonucleotides that are mixed and annealed with the DNA memories. In Baum's original proposal[9], recall was implemented by exact Watson-Crick complementary hybridization. Though DNA oligonucleotides prefer to hybridize with exact Watson-Crick complements, in fact, many energetically favorable hybridizations can occur with secondary structure, non-Watson-Crick base pairing, or mismatches. Therefore, the associative mechanism in a DNA-based memory could be implemented by favorable hybridizations between oligonucleotides that are near, but not exact, Watson-Crick complements. There have been several attempts at memory implementations[10, 11]. In addition, learning has been explored as a model for DNA computations[12, 13], but not as a storage mechanism for DNA-based memories.

In this work, the DNA computer is a laboratory protocol that through DNA-to-DNA reactions (hybridizations), first, stores molecules in a DNA-based memory, and then, matches new input to the stored molecules based upon sequence content. The storage procedure is called "learning" [14] because it acquires information from examples (the input DNA), and without explicit programming. The uniqueness of this design is the use of a learning protocol to store sequences in the memory, and recall through potential non-Watson-Crick hybridizations.

The DNA memory is an application that has the advantages of DNA computing. For example, new sequences are learned and stored ones recalled in one massively parallel step. In addition, the memory uses some of DNA computing's disadvantages, namely the imprecision of matching (the hybridization reaction), to advantage. Memory recall is implemented by degree of annealing between new input DNAs and the memory sequences, thus providing a technique for recognizing patterns based upon similarities in content. In addition, random interactions between DNA oligonucleotides in the form of inexact hybridizations are exploited for learning new input through a massive sampling of all possible subsequences of a given length.

In this paper, the design of the learning and recall protocols are given. Then, experimental results are presented that show that the learning protocol stores sequences as expected, that similar sequences are matched, and dissimilar se-

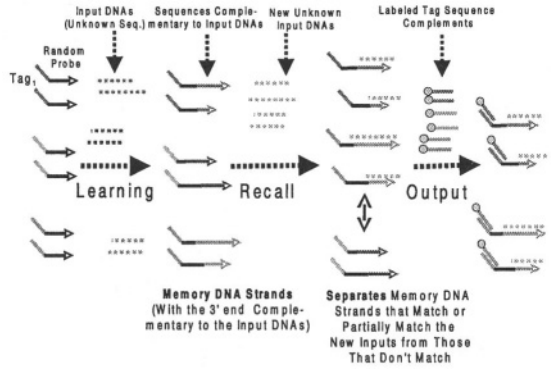


Fig. 1. A DNA-based associative memory. The memory strands are composed of memory specific sequences and tag sequences that are used for output

quences differentiated. In addition, the protocols are shown to learn and resolve fine differences among genomic amounts of DNA. It remains to be seen if the memory might be applied to the type of problem for which the Adleman-Lipton scheme is designed, but would seem to have potential for applications that require large amounts of storage density, an *in vitro* storage mechanism, and pattern matching by associative recall. Examples would include recognition of changes in environmental samples of genomic DNA, patterns in gene expression, and text storage and semantic retrieval.

2 Memory Overview

A schematic of the memory is shown in Figure 1. The initial sequences in the memory are a set of non-crosshybridizing tag sequences to which random sequences are appended during synthesis. With simple and common recombinant DNA operations, such as PCR, exonuclease digestion, and bead extraction, the memory learns complementary portions of input DNA sequences. Subsequently, the memory stands can be used to recall those sequences, or sequences that are close under hybridization affinity. The tag sequences are designed to be independent of each other in that they will not hybridize, and can be used for output by hybridizing to their complements on an array, or by biotin-avidin bead separation. In addition, the tag sequences could undergo additional processing, implementing a DNA^{2}DNA computation, as in [15].

3 Learning and Recall Protocols

The protocol for learning is described in Figure 2. Initially, the memory strands consist of a tag sequence with Biotin attached and short random probe sequences (20-mers). The input DNA, which the memory is to learn, is mixed with the tag

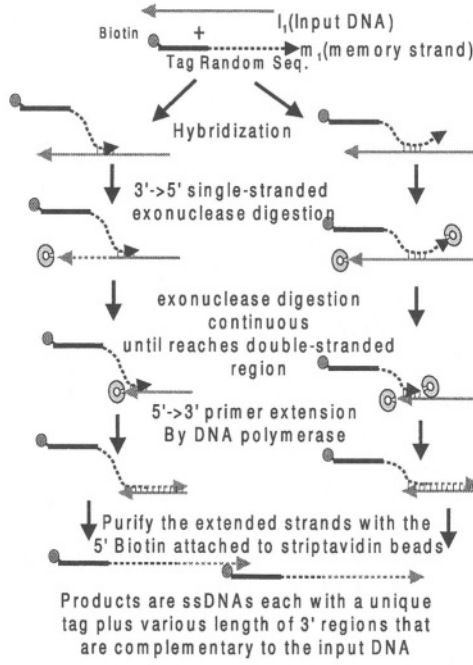


Fig. 2. Learning Protocol

plus probes. The probes will hybridize at random locations on the input DNA. A 3′ to 5′ exonuclease digests probe and input strands from the 3′ end until a double-stranded region is encountered. Then, a 5′ to 3′ extension by DNA polymerase is done. The extended memory strands, tag plus extended Watson-Crick complement of input, are separated from the input by the 5′ biotin attached to streptavidin beads. The products are single-stranded DNAs with a unique tag attached to random length 3′ regions that are complementary to the input DNA. To learn additional inputs, the process is repeated with a different tag. Therefore, up to the limit of the number of independent tag sequences, the DNA memory can learn and store DNA sequence information. For recall (Figure 3), unknown input is exposed to the different memory strands. The input will hybridize to memory sequences that are close to its Watson-Crick complement. The specific memory that is recalled can be determined from the tag that has the highest concentration of hybridized input.

4 Experimental Results

In the experiments, the goal was to test and verify the basic capabilities of the memory, which include learning of input DNA sequences, recall of the learned sequences, differentiation of very different sets of sequences, and generalization

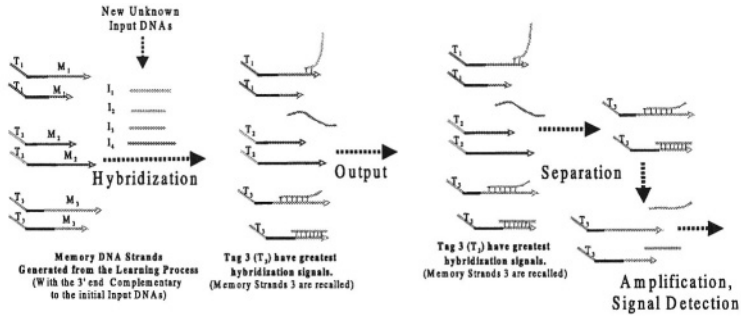


Fig. 3. Recall Protocol. I_1 through I_4 are input strands, M_1-M_3 are learned memory sequences, and T_1-T_3 are tags. Inputs are mixed with memory, and hybridize with the memory sequences M_1-M_3. Double stranded sequences are separated and amplified. The tag with largest concentration is memory that is recalled, in this case memory 3

to input DNA that is close, but not identical, to the learned sequences. In addition, experiments were done to test the sensitivity of the recall protocol, and to determine the extent to which the random probes cover large DNA input spaces.

Two plasmids, pBluescript (Input 1) and ϕx 174 (Input 2), were selected as the inputs for initial training. These plasmids were chosen because they have very different sequences, and thus, can be the basis for two unrelated memories. After digestion, the starting sets of input DNAs are between 50 and 200 bases long. In Figure 4-LEFT, a gel is shown with 4 test sets of DNA. The original plasmids after digestion, pBluescript and ϕx 174 are in lanes 2 and 4, respectively. In lane 1 is the size ladder, which is actually similar to pBluescript, and in lane 3 is a plasmid that shares an ampicillin resistant gene with pBluescript. The starting DNAs for the learning protocol were two distinct, non-crosshybridizing tag sequences of length 20 bp that had 20 bases of random DNA appended to them, for a total starting length of 40 bp. Using the learning protocol, Memory 1 strands were trained on pBluescript, and Memory 2 strands on ϕx 174. A denaturing gel (Figure 4-CENTER LEFT) shows different distributions for the learned sequences for the two memories, and successful extensions of between 60 and 100 bases. This indicates that the learning protocol is successful at randomly sampling the input space of DNA, creating a Watson-Crick complement of the inputs, and polishing the 3′ ends.

Next, the capabilities of the recall procedure was tested. In the blots to follow, the original plasmids, pBluescript and ϕx 174 are in lanes 2 and 4, respectively. In lane 1 is the size ladder, which is actually similar to pBluescript, and in lane 3 is a plasmid that shares an ampicillin resistant gene with pBluescript. In Figure 4-CENTER RIGHT, the Memory 1 DNA was radioactively labeled, and used as a probe in Southern blot with the 4 test sets. As seen, Memory 1 hybridized to the DNA in lanes 1-3, thus, recalling the DNA on which it was trained (lane 2), and two sets of DNA (lanes 1 and 3) which are similar, but not identical, to the training set. In addition, there was no hybridization, and

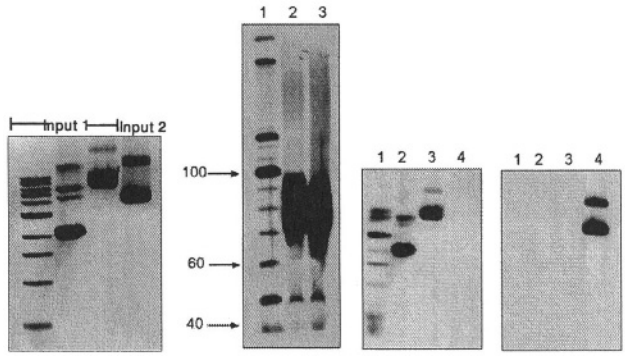


Fig. 4. LEFT: Gel showing sets of DNA for testing the recall protocol. Lane 1 is a ladder that is similar to input 1 (pBluescript, lane 2). Lane 3 is a plasmid that is 80% similar to pBluescript. Lane 4 is input 2 (ϕx 174). CENTER LEFT: Denaturing gel of learned DNA. Lane 1 is a standard ladder. Lane 2 are the sequences learned from input 1 (pBluescript), and lane 3 are the sequences learned from input 2 (ϕ x 173). Lanes 2 and 3 show a different distribution, as expected for different starting material. Also, lengths between 60 and 100 base pairs confirm that the learning protocol copied and polished input DNA as planned. CENTER RIGHT: Southern blot with Memory 1, which was trained on input 1, as radioactively labeled probe. Those sets that are similar (lanes 1-3) are recalled, but not the dissimilar set (lane 4). RIGHT: Southern blot with Memory 2, which was trained on input 2, as radioactively labeled probe. Input 2 is recalled (lane 4), but not dissimilar DNA (lanes 1-3)

thus, no recall of the very different set of DNA in lane 4. In Figure 4-RIGHT, the Memory 2 DNA was radioactively labeled, and used as a probe in Southern blot with the 4 test sets. In this case, only the DNA on which it was trained was recalled. Therefore, the ability of the DNA to recall through hybridization related DNA, and to differentiate between unrelated DNA is confirmed. In addition, in Figure 4, even though the DNA in lanes 2 and 3 share about 1kb of sequence, there is a detectable difference in their blot, thus giving a basis to detect related, but different sets of DNA.

Sensitivity of the technique was also investigated. Varying amounts of pBluescript were added to a background of ϕx 174. As seen in Figure 5, the technique was able to detect target DNA present in a concentration 1% of the background. In addition, a test was done to measure the ability of the starting random probes to cover the input space. Both Memory 1 and Memory 2 were trained on the combined input of pBluescript and ϕx 174. Then, as before, the blots of the inputs with Memory 1 and Memory 2 were done. As seen in Figure 6, the outputs were essentially identical, indicating the ability of the initial randomness to adequately cover the combined input space of approximately 8000 bp.

In addition, a test was done to measure the ability of the starting random probes to cover a large input space. Thus, the genome of *E. coli* ($\approx$ 5 million base pairs (bp)) was learned, and adequately recalled (Figure 7). In addition, the

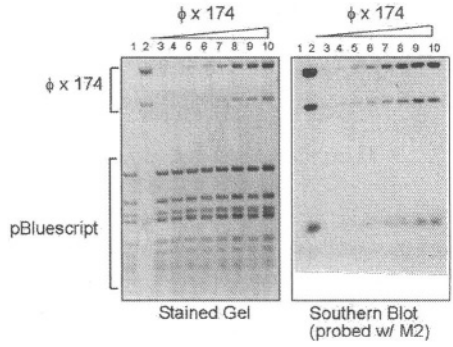


Fig. 5. The left panel is an agarose gel stained with ethidium bromide. Lane 1 contains 14 fragments (from ~ 100 bp to 600 bp) of pBluescript plasmid (1 ug; ~ 3 kb) digested with restriction enzyme Hpa II. Lane 2 contains 5 fragments (from ~ 200 bp to 1.7 and 2kb) of ϕx 174 plasmid (1 μg; $\sim$ 5 kb) digested with Hpa II. Each of Lanes 3 to 10, contains a fixed amount of pBluescript (1 μg) with increasing amount of ϕx 174 (10 ng, 20 ng, 50 ng, 100 ng, 200 ng, 400 ng, 600 ng, and 800 ng). The right panel is the same gel been blotted to nitrocellulose membrane and probed with Memory strand 2. It shows that even with only 1% of ϕx 174 (10 ng) present in pBluescript (1ug), it can still be detected

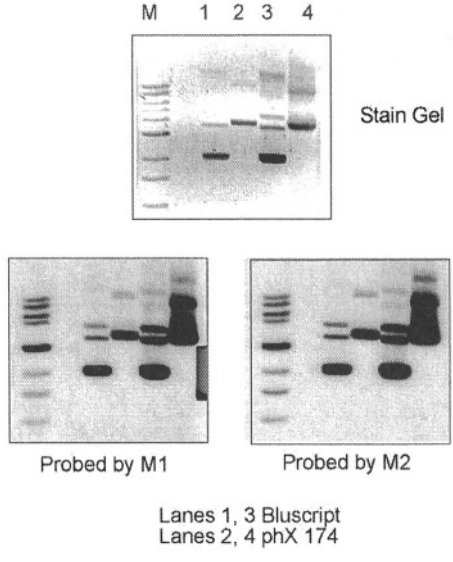


Fig. 6. The ability of learning protocol to cover two input spaces. The top stained gel shows the randomly digested inputs. Below, the Southern blots of Memory 1 (left) and Memory 2 (right) are very similar, which indicates that the initial randomness is more than adequate to learn the combined input space

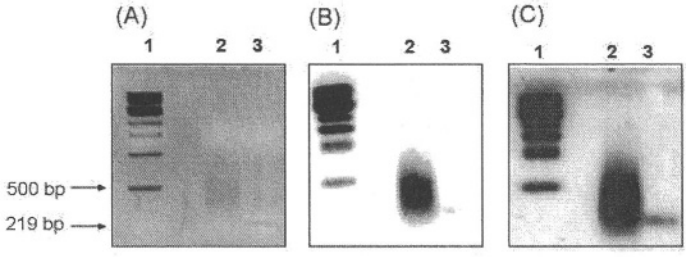
E. coli genome was learned with an additional 219 bp fragment of DNA from ϕx 174. The results are shown in Figure 7. After Southern blotting, the recall was able to distinguish the 219 bp piece from among the approximately 5 million bp of the *E. coli* genome, showing the capability for an adequate level of resolution.

5 Discussion

In neural network models of the brain, learning proceeds by localized adjustment of energetic interactions between units and deepening of energy minima corresponding to stored patterns. Memories are retrieved by relaxation to these energy minima. Similarly, the DNA memory learns by matching initial sequences to those in the input that produce the most energetically favorable hybridization, and then, strengthens that attraction by extending the initial matching sequence. Likewise, recall proceeds by hybridization to the stored sequence that produces the largest energy relaxation.

The memory has the potential to learn large DNA sequences. Learning from examples is taken in the sense of Valiant [14] in that the ability to recognize sets of input DNA is acquired without explicit programming. The DNA string learning problem was defined by Jiang and Li[16]. The task is to learn a set of strings (DNA molecules) over the alphabet $\{A, G, C, T\}$. This corresponds to the set of DNA input molecules to the learning protocol. The positive examples for a string i are its substrings; the negative examples are strings that are not substrings of i. The examples are sampled randomly. In the DNA memory, the positive and negative examples are determined by whether the random probes are Watson-Crick complementary to substrings of the input or not. Under the assumptions that the learning algorithm is polynomial in the size of the input, the number of examples needed to learn a DNA string can be taken as $O(n \log n/\sigma)$, where n is the length of the input string and $0 < \sigma < 1$ is the error probability[16]. If we take $\sigma = 1/n$, then, that means with 20-mer probes, the DNA memory can learn a sequence with about 10^6 bp. This estimate agrees with the *E. coli* results.

Furthermore, according to Cover's theorem on the separability of patterns[17], complex patterns that are projected by nonlinear functions into a high dimensional space can then by linearly separated. The biological memory demonstrated here can be understood in terms of Cover's theory, and thus, is similar to radial basis function classifiers[18]. The input for the biological memory is the set of input DNA, I, and the memory strands, m, are complementary copies of randomly sampled subsequences in the training set. The nonlinear mappings, $f(i \in I)$, to a high-dimensional space are implemented by hybridization to the learned memory strands. Input and memory DNAs hybridize with a probability determined by their hybridization energy, which is related to their degree of Watson-Crick complementarity. In the nearest-neighbor model of DNA duplex thermal stability[19], the energy of hybridization is given by the sum over the nearest-neighbor pairs in the duplex sequence weighted by the pair stacking energies. The probability of hybridization, P_H, to a particular memory, $m \in M$,



(A) Shown is a stain gel of E. coli genomic DNA digested with DNaseI (lane 2), and a 219 bp DNA fragment from ϕx 174 (lane 3). Lane 1 is the markers.
(B) Southern blot of gel shown in (A), probed by M1 which has been trained with E. coli genomic DNA as input.
(C) Southern blot of gel shown in (A), probed by M2 which has been trained with E. coli genomic DNA plus the 219 bp ϕx 174 DNA as input.

Fig. 7. The *E. coli* genome was digested into smaller pieces and learned as memory 1. Then, a 219 bp fragment of ϕ X 174 was added to the digested *E. coli* and learned as memory 2. Recall by blotting showed that memory 1 (*E. coli* alone) was not able to distinguish the ϕx 174 fragment, while learning with the fragment present (memory 2) was. Thus, the learning and recall protocols were able to distinguish a 219 bp sequence from approximately 5 million bp in *E. coli*

is then,

$$P_{\mathrm{H}} = \frac{\sum_{i \in I} \exp -\Delta G_{im}/RT}{Z}, \tag{1}$$

where R is the gas constant, T is the absolute temperature, Z is the partition function, and the sum is taken over the input DNAs I. The dependence on the input is contained in the exponential, and specifically, in the pairwise free energy of hybridization ΔG_{im}, between input and memory DNAs. Therefore, the mapping from input DNAs to memory DNAs is nonlinear and probabilistic. The output of the memory is proportional to the number of hybridized input-memory DNA strands associated with each tag. For instance, if the complements of the tags were affixed to an oligonucleotide chip, then, the output, as sensed optically, would be a linear function of the number of input DNA hybridized to the memory strands associated with a particular tag. The output could be thresholded to produce a dichotomy of the input space into two regions, for example, like pBluescript or not. If there are $|M|$ memory strands, Cover's theory suggests that the separating capacity of the dichotomy will be $2|M|$[17]. The number of memory strands is determined by the initial randomness associated with the tag and the amount of matching between the training DNA and the random probes. The initial randomness can be assumed to uniformly sample the input space, and for the concentrations typically used and sequences of length 20, may be assumed to contain every possible 20-mer sequence, Thus, theoretically, $M = 4^{20} \approx 10^{12}$. The human brain contains approximately $10^{11} - 10^{14}$ neurons. Therefore, assuming all the initial sequence randomness is incorporated into memory strands, the biological memory has a separating capacity on the order

of the human brain. It is the large amount of initial random sequence attached to the tags that gives the DNA memory its power to learn and separate input sequence patterns in a massively parallel search.

A potential application of the memory might be *in vitro* classification of gene expression. In this case, the memory would be trained on messenger RNA, or its DNA copy, under different conditions. For example, one memory would be trained on samples from healthy cells, while another from diseased cells. Then, new input could be classified as healthy or diseased by tag output. This would be a simple, inexpensive, and quick alternative to gene chips. Currently, many gene expression studies are done using DNA arrays to which every possible gene is affixed. Gene expression patterns correspond to the relative hybridization of different areas on the chip, and then, data analysis, and pattern recognition and classification is done with a conventional computer. If the work of pattern classification could be done *in vitro,* then, the huge potential separating capacity of the biological memory would represent an advantage over the conventional computer. Also, by separating the memory strands into tag and input-specific sequences, DNA can be learned and sensed without prior knowledge or explicit sequencing. This is a capability that a conventional computer or gene chip does not have. It works because the learning protocol does not require any knowledge of the input sequence, and the output on the tags, similarly to a universal DNA chip[20], is also independent of input sequences. By training on microbial DNA from an environment and then, periodically sampling the environment and matching with the stored strands, the memory could be used as sensor to detect change in the micro-organisms that are present. In addition, with appropriate encoding of non-biological data into DNA, the memory could be used to store large databases of information that could be semantically searched with the associative recall mechanisms. This memory could be exceptionally large. For instance, output could be represented by optical level on tag hybridization to a DNA chip. If the optical systems could resolve 10 levels of intensity on d tag sequences, then, the capacity of the memory would be 10^d. Up to 4000 independent tags have been designed [6], which would give a memory with the capability to produce potentially 10^{4000} outputs. Of course not all of these would represent individual memories, but it does give an idea of the potential storage capacity.

6 Conclusion

An advantage of DNA as a computational medium is its vast information storage density, with theoretically, 10^{21} bits in a gram[21]. The DNA memory theoretically uses a large proportion of the sequence space during the learning protocol, and its potential capacity reflects the vastness of that space. Another advantage is the massive parallelism of the DNA reactions, which is responsible for the potential speed-ups in DNAC. Programming and controlling that many simultaneous reactions, however, has been problematic, and thus, the trend in DNAC has been to allow the computation to proceed by self-assembly[22], in which simple rules direct the dynamic evolution of the computation. So far, the number

of self-assembly rules has been small, and to date, DNAC has not fully utilized the vast space of DNA sequences (4^n where n is the length). Because of the way the learning and recall have been implemented, precise control of the reactions is not necessary to either learn a sequence, or recall a sequence from the memory. In the DNA memory, sequence design is only an issue for the tag sequences, and there can be a limited number of those, which is within the capability of current design programs[6]. DNAC applied to biology has also been proposed[15, 23]. This would seem to be a natural application area. The learning protocol provides an simple and immediate method to incorporate biological sequences into a content-addressable memory. In addition, the classification capability of the memory is enormous, with a potential advantage over conventional techniques. The experimental results show that the proposed protocols work, and provide a simple, efficient, and practical input/output mechanism for a DNA-based memory. These capabilities are achieved without explicit programming of the learning and recall processes, but instead, are based upon *in vitro* adaptation of DNA sequences and the energetics of the hybridization reaction.

References

[1] Adleman, L. M.: Molecular computation of solutions to combinatorial problems. Science **266** (1994) 1021–1024

[2] Lipton, R. J.: DNA solution of hard computational problems. Science **268** (1995) 542–545

[3] Braich, R. S., Chelyapov, N., Johnson, C., Rothemund, P. W. K., Adleman, L.: Solution of a 20-variable 3-sat problem on a dna computer. Science **296** (2002) 499–502

[4] Deaton, R., Murphy, R. C., Garzon, M., Pranceschetti, D.R., Stevens Jr., S.E.: Good encodings for DNA-based solutions to combinatorial problems. In Landweber, L.F., Baum, E.B., eds.: DNA Based Computers II. Volume 44., Providence, RI, DIMACS, American Mathematical Society (1998) 247–258 DIMACS Workshop, Princeton, NJ, June 10-12, 1996.

[5] Marathe, A., Condon, A.E., Corn, R.M.: On combinatorial DNA word design. [25] 75–90 DIMACS Workshop, Massachusetts Institute of Technology, Cambridge, MA.

[6] Deaton, R., Chen, J., Bi, H., Rose, J. A.: A software tool for generating noncrosshybridizing libraries of DNA oligonucleotides. In: Preliminary Proceedings of the Eighth Annual Meeting on DNA Based Computers, Sapporo, Japan, Hokkaido University Japan (2002) 211–220

[7] Brennenman, A., Condon, A.E.: Strand design for bio-molecular computation. `http://www.cs.ubc.ca/~condon/papers/wordsurvey.ps` (2001)

[8] Hartmanis, J.: On the weight of computations. Bulletin of the European Association for Theoretical Computer Science **55** (1995) 136–138

[9] Baum, E.: Building an associative memory vastly larger than the brain. Science **268** (1995) 583–585

[10] Mills, A. P., Yurke, B., Platzman, P. M.: DNA analog vector algebra and physical constraints on large-scale DNA-based neural network computation. [25] 65–74 DIMACS Workshop, Massachusetts Institute of Technology, Cambridge, MA.

[11] Reif, J. H., LaBean, T. H., Pirrung, M., Rana, V. S., Guo, B., Kingsford, C., Wickham, G.S.: Experimental construction of very large scale DNA databases with associative search capability. In Jonoska, N., Seeman, N. C., eds.: DNA Computing: 7th International Meeting on DNA-Based Computers, Berlin, University of South Florida, Tampa FL, Springer-Verlag (2002) 231–247 Lecture Notes in Computer Science 2340.
[12] Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., Yokoyama, S.: Towards parallel evaluation and learning of Boolean μ-formulas with molecules. [24] 57–72 DIMACS Workshop, Philadelphia, PA.
[13] Sakakibara, Y.: Solving computational learning problems with boolean formulae on DNA computers. In Condon, A., Rozenberg, G., eds.: DNA Computing: 6th International Meeting on DNA-Based Computers, Berlin, Leiden University, Leiden, NE, Springer-Verlag (2001) 220–230 Lecture Notes in Computer Science 2052.
[14] Valiant, L. G.: A theory of the learnable. Communications of the ACM 27 (1984) 1134–1142
[15] Landweber, L., Lipton, R. J.: DNA^2DNA computations: A potential "Killer App". [24] 161–172 DIMACS Workshop, Philadelphia, PA.
[16] Jiang, T., Li, M.: DNA sequencing and learning. Math. Syst. Theory **26** (1996) 387–405
[17] Cover, T. M.: Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Transactions on Electronic Computers **14** (1965) 326–334
[18] Haykin, S.: Neural Networks: A Comprehensive Foundation. MacMillan College Publishing Co., New York (1994)
[19] SantaLucia, Jr., J.: A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. Proc. Natl. Acad. Sci. **95** (1998) 1460–1465
[20] Brenner, S.: Methods for sorting polynucleotides using oligonucleotide tags. US Patent 5604097 (1997)
[21] Reif, J.H.: Successes and challenges. Science **296** (2002) 478–479
[22] Winfree, E., Liu, F., Wenzler, L. A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature **394** (1998) 539–544
[23] Normile, D.: DNA-based computer takes aim at genes. Science **295** (2002) 951
[24] Rubin, H., Wood, D.H., eds.: DNA Based Computers III, Providence, RI, DIMACS, American Mathematical Society (1997) DIMACS Workshop, Philadelphia, PA.
[25] Winfree, E., Gifford, D.K., eds.: DNA Based Computers V, Providence, RI, DIMACS, American Mathematical Society (1999) DIMACS Workshop, Massachusetts Institute of Technology, Cambridge, MA.

Efficiency and Reliability of Semantic Retrieval in DNA-Based Memories

Max H. Garzon, Kiran Bobba, and Andrew Neel

Computer Science, The University of Memphis
mgarzon@memphis.edu

Abstract. Associative memories based on DNA-affinity have been proposed. Here, the efficiency, reliability, and semantic capability for associative retrieval of three models of a DNA-based memory are quantified and compared to current conventional methods. In affinity-based memories[l], retrievals and deletions under stringent conditions occur reliably (98%) within very short times (100 milliseconds), regardless of the degree of stringency of the recall or the number of simultaneous queries in the input. In a more sophisticated type of DNA-based memory B proposed and experimentally verified by Chen et al. [2] with three genomes, the sensitivity of the discrimination ability remains unchanged when used on a library of 18 plasmids in the range of l-4kbps and does appear to grow exponentially with the number of library strands used, even under simultaneous multiple queries in the same input. Finally, using a new type of memory compaction mechanism for data mining in vitro, DNA-based semantic retrieval compares favorably with statistically-based Latent Semantic Analysis (LSA), one of the best performers for semantic associative-based retrieval on text corpora.

Keywords: DNA-based memories, pattern classification, data compaction, semantic retrieval, optimal concentration, data mining *in vitro.*

1 Introduction

Potential applications of DNA include the creation of memories that can store very large amounts of data and fit into minuscule spaces [1,9,10]. The astonishing capacity of DNA (over millionfold compared to current electronic media) and the advances in recombinant biotechnology to manipulate DNA *in vitro* in the last 20 years, make this approach attractive and potentially very promising. Despite much work in the field, however, difficulties still abound in bringing these applications to fruition due to several reasons. First, inherent difficulties exist in orchestrating a large number of individual molecules to perform a variety of functions in the environment of test tubes, where the complex machinery of the living cell is no longer present to organize and control the numerous errors pulling computations by molecular populations away from their intended targets. Second, the required input procedures require fabrication of very large numbers of DNA molecules encoding possibly abiotic information. Third, reliable output may require detection of minuscule under-threshold amounts of reaction products to perform query operations [9,10,11].

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 157-169, 2004.

In this paper, we initiate a quantitative study of the efficiency and actual associative recall capacity of memories *in vitro.* The idea of using DNA to create large associative memories goes back to Baum [1], where he proposed to use DNA as the basic medium for content-addressable storage of information so that retrieval could be accomplished using the basic mechanism of DNA hybridization affinity. Content is to be encoded in single stranded molecules in solution (or their complements.) Queries can be obtained by dropping in the tube a DNA primer Watson-Crick complement of the (partial) information known about a particular record using the same coding scheme as in the original memory, appropriately marked (e.g., using streptavidin beads, or fluorescent tags.) After appropriate reaction times have been allowed for hybridization to take effect, retrieval is completed of any resulting double strands by any one of several extraction methods (e.g., sequencing or optical readout in gel electrophoresis.) As pointed out by Baum [1], and later Reif & LaBean [9], many questions need to be addressed before a large associative memory of this type approaching the capacity of the brain can be regarded as feasible, let alone actually built.

Further methods were proposed in [9,10] for input/output from/to databases represented in wet DNA (such as genomic information obtained from DNA-chip optical readouts, or synthesis of strands based on such output). The proposed hybrid methods in [9], however, require major pre-processing of the entire database contents (through clustering and vector quantization) and post-processing to complete the retrieval by the DNA memory (based on the identification of clusters centers.) This is a real limitation when the presumed database approaches the expected sizes to be an interesting challenge to conventional databases, or when the data already exists in wet DNA because of the prohibitive cost of the transduction process to and from electronics. In [10], experimental methods are explored for improving the reliability and sensitivity of retrievals of DNA-based memories once the query has been completed. Inherent issues in the retrieval *per se,* such as their reliability (e.g., false negatives not addressable by good encodings) and the appropriate concentrations for optimal retrieval times and error rates *in vitro* remain unclear. Furthermore, the discrimination ability of the associative nature of the retrieval, which must be the case for DNA due to the nature of their storage in solution or solid support, remains an unexplored issue. As the World Wide Web has made patently clear, making available enormous amount of data is only the first step; proper organization and tools are required to mine them and extract useful information and knowledge, something at which brains are particularly efficient. In particular, the necessary step of a comparison with the conventional methods for content-addressable retrieval has not been, to our knowledge, properly quantified objectively for DNA memories. How much can a DNA-based memory actually "know" and/or "learn"?

In this paper, we present an assessment of the efficiency and reliability of queries in three distinct models of DNA-based memories, Baum's memory B, a genomic DNA-based memory CDW, proposed by Chen et al. [2], for identification and recognition, and a new proposed type of memory compaction mechanism P, which are described in Section 2. In Section 3, we describe the goal of the experiments and the design to collect appropriate data. In Section 4.1, we present estimates of the reliability and efficiency of basic retrieval and delete operations in all affinity-based

memories. In section 4.2, we present results concerning their ability to compact, categorize, and discriminate inputs. In section 4.3, we present a new application of the selection protocol of Deaton et al. [3] and produce a preliminary assessment of the ability of DNA to summarize, compact, and extract useful information from large data sets. Based on these experiments, we conclude with a preliminary assessment of the quality for semantic processing of memories CDW and P, in comparison with the best methods known in conventional computing.

2 DNA-Based Memories and Databases

The first model of associative memories is essentially that suggested by Baum [1] and is described above in the Introduction. A library is a test tube containing a large diverse population of noncrosshybridizing oligonucleotides (up to 150bps or so) that either encode the contents directly, or serve as labels to index the information of particular records. Preliminary experiments confirming the feasibility of creating such large libraries *in vitro* were reported in [3]. Potential applications and methods to preprocess information to store in these libraries have been discussed in [9,2]. Our focus here is rather on the basic operations to manipulate these libraries for the purpose of exploiting the associative recall capability and building a full file system for a molecular computer.

The second is a novel and more sophisticated memory introduced by Chen et al. [3]. It is more appropriate for the storage and recognition of very large strands of DNA (e.g., full genomes of biological organisms, or large corpora of text data.) A record (genome) is stored by a compaction procedure described as *learning* in [2]. A genome G is first reduced to a *set* of short strands obtained by shearing it into a number of relatively small residues (in the order of 100 to 500 bps), obtained, for example, by the use of restriction enzymes; equivalently, the genomes can be put in without shearing because of the digestion step below. They are placed in a tube containing a library of double-stranded *tags* as used in a Baum's type memory, but extended by adding random segments of about residue length. The residues are allowed to attach to the random extensions, possibly leaving some bulges. After digesting with exonuclease to remove the bulges, extending with polymerase (as pruned by the residues), and removing the extended residues to single strand their tagged complements, a set of strands is obtained representing the original genome G, each attached to a common tag. This representation can be visualized in a *signature* that shows the hybridization product of G to each of the tags in the library using gel electrophoresis. An example of the representation in linear form for the plasmid *Vibrio Cholera* is shown in Fig. 1 (blue, more below.) This procedure is iterated and the library is trained on all genomes to be contained in the memory. Further details and experimental confirmation of the feasibility of this procedure can be found in [2].

The device is to be utilized for recognition and comparison of given genomes. Identifying or recognizing the presence of an unknown genome X (the *query*) can be done in two ways, First, we can produce X's signature and compare it directly with the database of known genomes. This requires maintaining a database base of all known signatures, which may be prohibitively expensive if their population is too large. Alternatively, X can be sheared as it is learned, and the complementary pieces

allowed to hybridize with the trained memory to produce a second signature. Substantial differences in the original and new signature signal the presence of a new (perhaps contaminating) genome X, while (nearly) identical signatures indicate X was already present in the database. An example is shown in Fig. l(b), where a contaminating piece has been added to produce the yellow (white) signatures.

The third memory P is a mechanism to extract useful information (data mine, so to speak), a given DNA-based memory, and store it compactly in a memory of type B. The compaction mechanism is a refinement of the selection protocol described in [3] to generate large libraries of noncrosshybridizying DNA strands (more below.)

In order to address the issues discussed above, three experiments were designed and performed to evaluate each of the three memories, as described next.

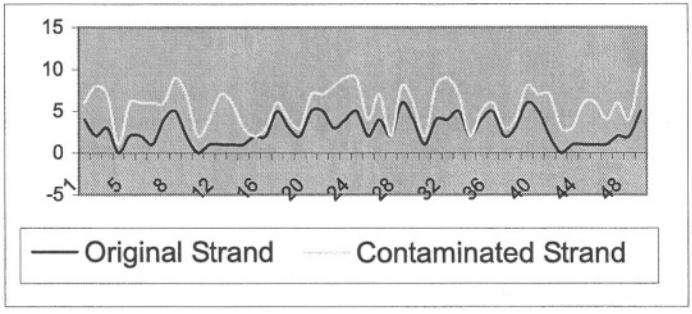


Fig. 1. (a) Signaturs of a compact representation of plasmid VibrioCholera (2560 bp) on a tag library of twenty 20-mers (blue); and (b) a small variant (yellow)

3 Experimental Design

The experiments described below were run a number of times. The results reported are averages of appropriate sets that total over 2000 simulations performed.

3.1 Virtual Test Tubes

Most of the experimental data used in this paper has been obtained by simulations in the virtual test tube *Edna* of Garzon et al. [6,7]. Recent results [5] show that these simulations produce results that closely resemble, and in many cases produce nearly identical results to, the protocols they simulate in wet tubes [5]. For example, Adleman's experiment has been experimentally reproduced and scaled in virtual test tubes with random graphs of up to 15 vertices while producing results correct with no probability of a false positive error and a probability of a false negative of at most 0.4% [7,5]. Virtual test tubes have also matched very well the results obtained in vitro by more elaborate and newer protocols, such as the selection protocol for a DNA library design of Deaton et Al. [3], used below. Therefore, there is good evidence that virtual test tubes provide a reasonable and reliable estimate of the events in wet tubes in the context of associative memories as well. The reader is referred to [7,6,5] for further details about *Edna.*

The interactions among objects in *Edna* represent chemical reactions by hybridization and ligation resulting in new objects such as dimers, duplexes, double strands, or more complicated complexes. They can result in one or both entities being destroyed and a new entity possibly being created. In our case, we wanted to allow the entities that matched to hybridize to each other to effect a retrieval, per Baum's design [1]. *Edna* simulates the reactions in successive *iterations.* One iteration moves the objects randomly in the tube's container (the RAM really) and updates their status according to the specified interactions with neighbor objects, based on proximity parameters that can be varied within the interactions. The hybridization reactions between strands were performed according to the *h*-measure [7] of hybridization likelihood. Hybridization was allowed if the *h*-distance was under a given threshold, which is the number of mismatches allowed (including frame-shifts) and so roughly codes for stringency in the reaction conditions. A threshold of zero enforces perfect matches in retrieval, whereas larger values permit more flexible and more associative retrievals. These requirements essentially ensured appropriate matches along the sections of the sample DNA X that affect the associative recall.

The efficiency of the protocols (in our case, retrievals) can be measured by counting the number of iterations necessary to complete the reactions or achieve the desired objective; alternatively, one can measure the wall clock time. The *number of iterations* taken until a match is found has the advantage of being indifferent to the speed of the machine(s) running the experiment. This intrinsic measure was used because one iteration is representative of a unit of real-time for experiments *in vitro.* The relationship between simulation time and equivalent reaction time has been discussed in [5]. Essentially, one iteration of the test tube corresponds to the reaction time of one hybridization event in the wet tube, which is of the order of one millisecond. However, it cannot be a complete picture because iterations will last longer as more entities are put in the simulation. For this reason, *processor time* (wall clock) tune was also measured. The wall clock time depends on the speed and power of the machine(s) running *Edna* and ranged anywhere from minutes to days for the single processors and 16 PC cluster that were used to run the experiments used below.

3.2 Libraries and Queries

We assume we have at our disposal a library of non-crosshybridizing strands representing the records in the databases. Well-chosen DNA word designs that will make this perfectly possible in large numbers of DNA strands directly, even in real test tubes, may be available in the near future [3]. The non-crosshybridizing property of the library strands will also ensure that retrievals will be essentially noise-free (no false positives), module the flexibility built into the retrieval parameters (here, *h*-distance.) A record may also contain an additional segment (perhaps double-stranded [2]) encoding supplementary information beyond the tag or segment actively used for associative recall, although this is immaterial for assumptions and results in this paper. The library is assumed to reside in solution in the test tube, where querying takes place.

When a query comes close enough to a library (probe) strand in the tube so that any hybridization between the two strands is possible, an encounter (which triggers a check for hybridization) is said to have occurred. The number of encounters can vary greatly depending directly on the concentration of queries and library strands. It would appear that higher concentrations reduce retrieval time, but this is only true to a point since results below show that too much concentration will interfere with the retrieval process. In other words, a large number of encounters may cause unnecessarily hybridization attempts that will slow down the simulation. Further, too many neighbor strands may hinder the movement of a probe or query strand in search of its match. Probing is considered complete when duplexes of the query copies have formed retrieval duplexes with all possible library strands that should be retrieved (perhaps none) according to the stringency of the retrieval (here based on an *h*-distance threshold.) In single queries with high stringency (perfect matches) or efficiency (based on single molecules [11]), querying can be halted when a successful hybridization occurs. Lesser stringency and multiple simultaneous queries require longer times to complete the query. How long is long enough to complete a query with high reliability under these conditions?

3.3 Test Libraries and Queries

The experiments used mostly a library consisting of the full set of 512 non-complementary 5-mer strands, although other libraries obtained through the software package developed based on the thermodynamic model of Deaton et al. [4] were also used with consistent results. The former case is desirable to benchmark retrieval performance since the library is saturated (maximum size) and retrieval times would be worst-case. The queries were chosen to be random queries of 5-mers. The stringency was maximum (*h*-distance 0), so exact matches were required. The experiment began by placing variable concentrations (number of copies) of the library and the queries into a tube of constant size. Once placed in the tube, the simulation begins. It stops when the first hybridization is detected. For the purposes of these experiments, there existed no error margin thus preventing close matches from hybridizing. Introduction of more flexible thresholds does not essentially affect the results of the experiments.

In the first experiment, we collected data to quantify the efficiency of the retrieval process (time, number of encounters, and attempted hybridizations) with single queries between related strands and its variance in hybridization attempts until successful hybridization. Three batches of runs were designed to determine the optimal concentrations with which the retrieval was both successful and efficient, as well as to determine the effect on retrieval times of multiple queries in a single input. The experiments were performed between 5 and 100 times each and the results averaged.

A second experiment was done for similar analysis of the signature-based memory using the genomes of 18 different plasmids as shown in Table 1. They were downloaded from the National Center for Biotechnology Information's database at *ftp://ftp.ncbi.nih.gov/genbank/gbbct6.seq.gz*.

A third experiment for semantic retrieval was done with a small corpus of text data used in curriculum scripts for *AutoTutor,* an autonomous tutoring system capable of natural dialog that performs at the level of unskilled human tutors for qualitative physics (http://www.autotutor.org/). The curriculum scripts were designed by an expert team of physicists and psychologists to provide AutoTutor with a repertoire of possible answers to be semantically matched to a student's contributions in tutoring sessions in natural language. The semantic matching is done using Latent Semantic Analysis (LSA), one of the best performing tools available for semantic comparisons of text in natural language [12] (comparable to the level of advanced student's comprehension in tests of English as a Second Language.) LSA is trained on a corpus (e.g., all paragraphs in a physics textbook) by creating a matrix of co-occurrences of significant words in the corpus (removing articles and other low content words) and compacting the corpus to the most significant dimensions (e.g., 300) using Singular Value Decomposition (SVD). Once trained, LSA provides a cosine value in the compact space in the low dimensional projections of two paragraphs given as query. We selected 9 queries on which LSA performs very well and 9 queries on which LSA performs poorly in the comparisons of semantic meaning. For comparison, the corpus of 1024 paragraphs was translated into a library of 1024 DNA sequences to create a DNA corpus. The corpus was then subjected to a compaction procedure (similar to SVD) using the PCR-based selection protocol of Deaton et al.[3]. This filtering procedure eliminated most of the strands and selected a *compacted corpus* of 28 strands that "summarize" the corpus. To compare two queries, the (normalized) *h*-distance[7] was returned between the two closest elements in the compacted corpus (rare ties were broken randomly). Perfect complements are at *h*-distance 0 (low); completely mismatched strands (such as a run of As and a run of Gs), is at distance 100% (high value). The answer to a given query was the closest *h*-distance match in the compacted memory.

Table 1. Plasmids used in the evaluation of the DNA-based genomic database

Nr/[Size]	Genome
1, 2 [1482,1560]	Uncultured alpha proteobacterium partial 16S rRNA gene, Zymomonas mobilis phosphoglyceromutase (pgm) gene
3, 4 [1503,1443]	Propionibacterium propionicus 16S rRNA gene, Planktothrix rubescens 16S rRNA gene
5, 6 [1412,1506]	X.autotrophicus 16S ribosomal RNA, Xanthomonas campestris pv
7, 8 [2042,1901]	R.faecitabidus gene for rarobacter protease I (RPI), Vibrio cholerae p-hydroxyphenylpyruvate dioxygenase (ppdA) and ORF2 protein genes
9, 10 [2587,2656]	Zooloea varnigera PHB polymerase gene, Yersinia pseudotuberculosis plasmid encoded yopH gene
11, 12 [2557,2529]	Photorhasdus luminescens LuxD gene, X.campestrisor pFN gene
13, 14 [3109,3153]	V.cholerae otnA gene, Z.mobilis tgt and lig genes encoding tRNA guanine transglycosylase & ligase
15, 16 [3212,2924]	Zymomonas Mobilis Phosphoglucose isomerase gene, S.carnosus gene for 23S rRNA
17, 18 [2908,3060]	X.campestris gumD gene, Pseudomonas aeruginosa 6-phosphogluconate dehydratase (edd) gene

4 Analysis of Results

4.1 Retrieval Efficiency in DNA-Based Memories

Figure 2(a) shows the results of the first experiment at various concentrations averaged over five runs. The most hybridization attempts occurred when the concentration of queries is between 50-60 copies and the concentration of library strands (probes) was between 20-30 copies. Figure 2(b) represents the variability (as measured by the standard deviation) of the experimental data. Although, there exists an abnormally high variance in some deviations in the population, most data points exhibited deviations less than 5000. Interestingly enough, the range of 50-60 query copies and 20-30 query (library) copies, in proportion of 3:1, simultaneously exhibits minimum variation.

Figure 3(a) shows the average retrieval times as measured in tube iterations. The number of iterations decreases as the number of queries and library strands increase, to a point. One might think at first that the highest available query and library concentrations are desirable. However, Fig. 3(a) indicates a diminishing return in that the number of hybridization attempts increases as the query and library concentrations increase. If the ranges of concentrations determined from Fig. 2(a) are used, the number of tube iterations for successful retrieval remains under 200. Fig. 3(b) shows only minimum variability once the optimal concentration has been achieved. The larger deviation at the lower concentrations can be accounted for by the highly randomized nature of the test tube simulation. These results on optimal concentration are consistent and further supported by comparison with the results in Fig. 2.

As a comparison, in a second batch of experiments with a smaller (much sparser) library of 64 32-mers obtained by a genetic algorithm [6], the same dependent measures were tested. The results (averaged over 100 runs) are similar, but are displayed in a different form below. In Fig. 4(a), the retrieval times ranged from nearly 0 through 5000 iterations. For low concentrations, retrieval times were very large and exhibited great variability. As the concentration of query strands exceeds a threshold of about 10, the retrieval times drop under 100 iterations, assuming a library strand concentration of about 10 strands.

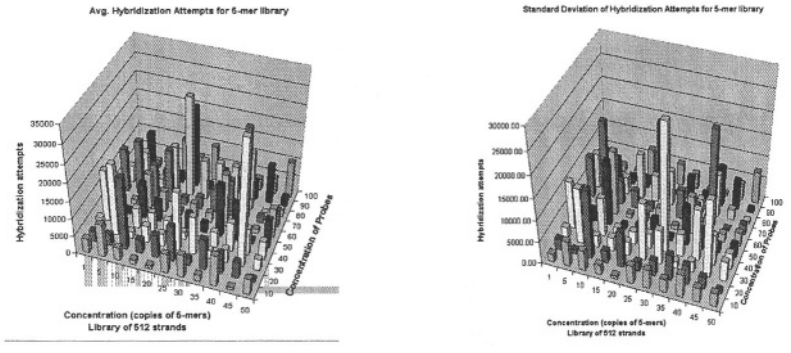


Fig. 2. (a) Retrieval difficulty (hybridization attempts) based on concentration; (b) Variability in retrieval difficulty (hybridization attempts) based on concentration

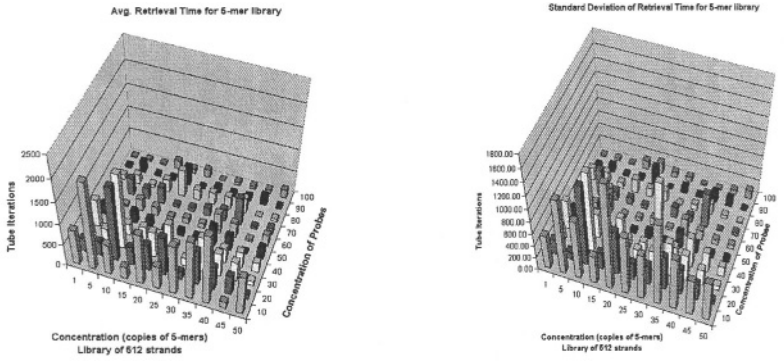


Fig. 3. (a) Retrieval times (number of iterations) based on concentration. (b) Variability of retrieval times (iterations) based on optimal concentration

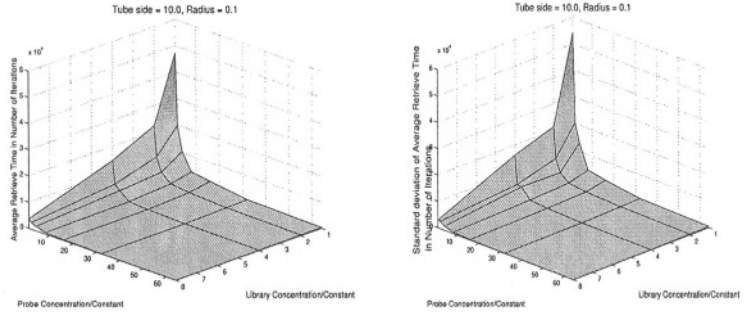


Fig. 4. (a) Retrieval times and optimal concentration on sparser library. (b) Deviation

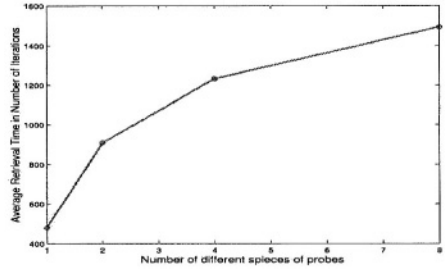


Fig. 5. Retrieval times (number of iterations) based on simultaneous multiple queries

Finally, Fig. 5 shows that the retrieval time increases only logarithmically with the number of (simultaneous) multiple queries in an input and tends to level off in the range within which queries don't interfere with one another.

In summary, these results permit an empirical estimate of optimal and retrieval times for queries in DNA associative memories. For a library of size N, an estimate of a good concentration of library strands for optimal retrieval time appears to be in the order of O(logN). Query strands require the same order, although probably about a

third of library concentration will suffice. The variability in the retrieval time is simultaneously optimized for optimal concentrations. Although not reported here in detail due to space constraints, similar phenomena were observed for multiple query inputs. We surmise that this holds true up to O(logN) simultaneous queries, past which queries begin to interfere with one another and cause a substantial increase in retrieval time. Based on benchmarks obtained by comparing simulations in Edna with wet tube experiments [5], we can estimate the actual retrieval time itself under optimal conditions to be in the order of 1/10 of a second for libraries in the range of 1 to 100 million strands in a wet tube (a low number compared to the capacity estimates in [2].)

It is worth noticing that similar results may be expected for memory updates. Adding a record is straightforward in DNA-based memories (assuming that the new record is noncrosshybridizing with the current memory), one can just combine it into the solution (perhaps through learning.) Deleting a record requires making sure that all copies of the records are retrieved (with full stringency for perfect recall) and expunged, which reduces deletion to the problem discussed in Section 4.1. Additional experiments were performed that verified this conclusion (results not shown.) The problem of adding new crosshybridizing records is of a different nature and was not directly addressed, although the results in Section 4.3 below shed some light on it.

4.2 Discrimination Sensitivity in DN A-Based Databases

Now we discuss the results for the DNA-based memory CDW of Chen et al. [3]. Fig. 6(b) shows the mean-difference between the two signatures. In order to evaluate the result, we need an objective measure of similarity to compare the original plasmid genomes. A thermodynamic model such as that in [4] would be ideal, but the strands are beyond the model's range (up to 150 bps or so) and the running time is prohibitive. Therefore, we used the *h*-distance[7] again as objective measure, shown in Fig. 6(a).

Most of the *h*-distances between the plasmids are in a narrow range of about 70% the size of the shorter length, so they are very different ("orthogonal") genomes. On the other hand, the mean distances between their signatures in Fig. 6(b), although smaller by comparison, remain high enough (not black) to differentiate two of them with as high reliability as with the measure using full genome information. Ambiguous genomes 12-17 are now distinct in signature, and only 9-16 are now ambiguous. This uniformly high correlation between the objective and test measures in Figs. 6(a) and (b) shows that despite the compaction, the discrimination ability of the learning procedure has preserved enough of the differences between the genomes for identification and recognition purposes in a memory of many genomes.

4.3 Semantic Sensitivity of DNA-Based Databases

Finally, Fig. 7 shows the result of the comparison between DNA-based and LSA-based semantic retrievals. Fig. 7(a) shows LSA performance (pairwise cosine indices of a selection of 32 of the 1024 strands is shown) on queries where code words are of such quality that a Baum-type DNA memory recalls perfectly with no errors (a

diagonal matrix, not shown), even under a very liberal recall condition. In the reciprocal test, DNA performs uniformly for all queries regardless of how good or bad LSA retrieves them in all three frames in Fig 7(b)-(d). Fig. 7(b) shows the matching without normalizing the *h*-distance. After normalization to account for different paragraph size, a high correlation became evident between the retrieval using (c) affinity to summary strands via *h*-distance of the queries to a corpus strand (without compaction), and (d) the affinity of the same queries to strands in the compacted corpus. All things considered, this is a remarkable performance, given that the corpus has shrunk to about 3% of the original and that we are dealing with semantic relationships involving high-level concepts (qualitative mechanics in physics.)

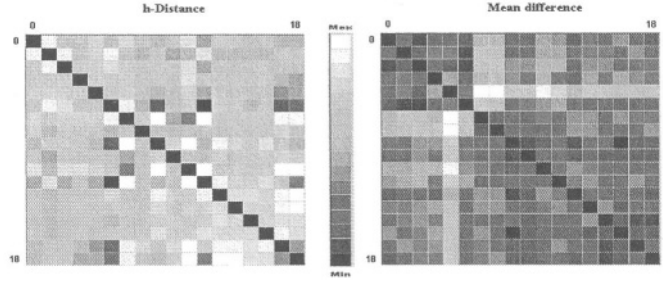


Fig. 6. Discrimination ability of memory CDW using mean distance between signatures. Black (light) colors show high (low, resp.) signature similarity, consistently with low affinity of original genomes. Only genome 6 is ambiguous and can be mistaken as genome 12, and even perhaps 17. In signature they can, however, be differentiated, although 9 and 16 are now ambiguous

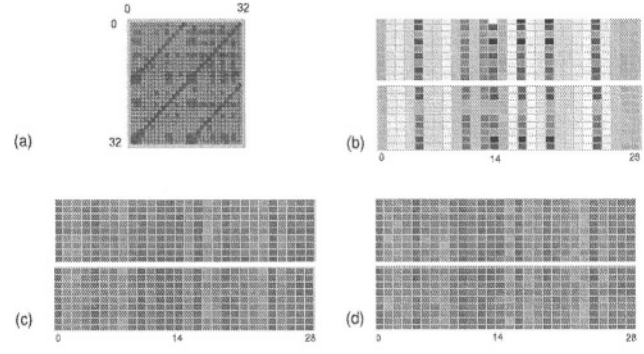


Fig. 7. Quality of semantic retrieval in memory P. Dark (light) colors indicate high (low, resp.) affinity and likely retrieval and vice versa. (a) LSA performance on best DNA queries and (others) h-distance: Nine LSA-good queries (top 9 rows) and nine LSA-bad queries (bottom 9 rows) using (b) closest DNA retrieval as control. The best match in (c) the corpus strand library and (d) the best match in the compacted corpus exhibit nearly identical patterns, i.e., DNA performs uniformly well on both type of queries regardless of LSA's performance

5 Summary and Conclusion

The reliability, efficiency, and compaction capabilities for semantic retrieval of two types DNA-based associative memories *in vitro,* as originally proposed by Baum[2] and Chen et al. [3], as well as new data mining procedure for DNA memories, have been quantitatively examined, through simulation of reactions *in silico* in a virtual test tube [7]. The results support the conclusion that there is a region of optimal concentrations for library and query strands to minimize retrieval time and avoid excessive concentrations (which tend to lengthen retrieval times) at about O(logN), where N is the size of the library. In addition, the retrieval time is highly variable depending on reactions conditions and queries, but tends to stabilize at optimal concentrations. Moreover, these results remain essentially unchanged for simultaneous multiple queries if they remain small compared to the library size, within the range of O(log N). Previous benchmarks of the virtual tube [5,7] provide a good level of confidence that these results extrapolate well to wet tubes with real DNA. Retrieval times *in vitro* can thus be estimated in the order of 1/10 of a second.

Furthermore, the results also show that data mining is very promising *in vitro* with DNA-based memories. They can compact and summarize information prior to retrieval (as must occur in a DNA-based genomic database) at least as well as the best algorithms available for text corpora (Latent Semantic Analysis (LSA) in terms of discrimination and associative recall capability. Therefore, further experiments with larger libraries *in vitro* are likely to exhibit a performance comparable to that of memory B on *pBlueScript, phix174,* and *E-coli* genomes described in [4], as well as to exhibit the intelligence required for a semantic DNA-based memory.

Acknowledgements

Thanks go to H. Chen, D. Jamisetti, K. Yallapu, and PhP. Penumatsa for their help with data. Supported by National Science Foundation grant QuBic/IEA-0130385. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] E. Baum, Building An Associative Memory Vastly Larger Than The Brain. *Science* **268** (1995), 583-585.

[2] J. Chen, R. Deaton, Y-Z. Wang. A DNA-based Memory with in vitro Learning and Associative Recall (2003). In: Proc. DNA9 (2003), Springer-Verlag Lecture Notes in Computer Science, these Proceedings.

[3] R. Deaton, J. Chen, H. Bi, M. Garzon, H. Rubin, D.H. Wood. A PCR-Based Protocol for In-Vitro Selection of Non-Crosshybridizing Oligonucleotides, In [8], 196-204.

[4] RJ. Deaton, J. Chen, H. Bi, J.A. Rose: A Software Tool for Generating Non-crosshybridizing Libraries of DNA Oligonucleotides. In [8], pp. 252-261.

[5] M. Garzon, D. Blain, K. Bobba, A. Neel, M. West, Self-Assembly of DNA-like structures *in silico.* In Journal of Genetic Programming and Evolvable Machines **4** (2003), 185-200.

[6] M. Garzon, Biomolecular Computation in silico. *Bull. of the European Assoc. For Theoretical Computer Science* EATCS 19(2003), 128-144.

[7] M. Garzon, C. Oehmen: Biomolecular Computation on Virtual Test Tubes, In: Proc. DNA7 (2001) Springer-Verlag Lecture Notes in Computer Science **2340** (2002), 117-128.

[8] M. Hagiya, A. Ohuchi (eds.) Proc. of DNA7, Hokkaido U, 2001. Springer-Verlag Lecture Notes in Computer Science **2568** (2002).

[9] J.H. Reif, T. LaBean. Computationally Inspired Biotechnologies: Improved DNA Synthesis and Associative Search Using Error-Correcting Codes and Vector Quantization. Proc. of the 6th International Workshop on Springer-Verlag Lecture Notes in Computer Science **2054**, 145-172.

[10] J.H. Reif, T. LaBean, M. Pirrung, V.S. Rana, B. Guo, C. Kingsford, G.S. Wickham. Experimental Construction of Very Large DNA Databases with Associative Search Capability. Proc. of DNA7, Springer-Verlag Lecture Notes in Computer Science **2340**, 231-247.

[11] K.A. Schmidt, C.V. Henkel, G. Rozenberg: DNA computing with single molecule detection. In Proc. of DNA7, Hokkaido U (2001), p. 336.

[12] T.K. Landauer, P.W. Foltz, D. Laham: Introduction to Latent Semantic Analysis. Discourse Processes **25** (1998), 259-284.

[13] Wetmur, J.G.: Physical Chemistry of Nucleic Acid Hybridization. In: Proceedings of the 3rd DIMACS Meeting on DNA Based Computers, University of Pennsylvania, June 1997.

Nearest-Neighbor Thermodynamics of DNA Sequences with Single Bulge Loop

Fumiaki Tanaka[1], Atsushi Kameda[2], Masahito Yamamoto[3], and Azuma Ohuchi[4]

[1] Graduate School of Engineering
Hokkaido University
North 13, West 8, Kita-ku, Sapporo 060-8628, Japan
fumiaki@dna-comp.org
http://ses3.complex.eng.hokudai.ac.jp/
[2] Japan Science and Technology Cooperation (JST)
Honmachi 4-1-8, Kawaguchi 332-0012, Japan
kameda@dna-comp.org
[3] PRESTO, Japan Science and Technology Cooperation (JST)
and Graduate School of Engineering, Hokkaido University
North 13, West 8, Kita-ku, Sapporo 060-8628, Japan
masahito@dna-comp.org
[4] CREST, Japan Science and Technology Cooperation (JST)
and Graduate School of Engineering, Hokkaido University
North 13, West 8, Kita-ku, Sapporo 060-8628, Japan
ohuchi@dna-comp.org

Abstract. Forty thermodynamic parameters were estimated for DNA duplexes with a single bulge loop. In DNA computing, sequences need to form wanted structures, not unwanted structures. To achieve this, we should design sequences with low free energy (ΔG°_{37}) in wanted structures and high free energy in unwanted structures. Conventional sequence design strategies have not prevented the formation of bulge loop structures completely. Estimation of the ΔG°_{37} of the bulge loop with the loop length from the chemical experimental data has not been enough to predict the ΔG°_{37} of the bulge loop structure. To investigate the effect of the type of bulged base and its flanking base pairs, we applied the nearest-neighbor model to DNA sequences with a single bulge loop. We also estimated the effect of loop position on the stability of a single bulge loop.

1 Introduction

In DNA computing, we need the sequences to form wanted structures, not unwanted structures. For this purpose, we should design sequences with low free energy (ΔG°_{37}) in wanted structures and high free energy in unwanted structures. Therefore, successful DNA computing requires as accurate a prediction of ΔG°_{37} for each DNA structure as possible.

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 170–179, 2004.

Conventional sequence design strategies do not completely prevent the formation of bulge loop structures [12][13]. The ΔG°_{37} of the bulge loop has been estimated with the loop length from chemical experimental data [1], but the stability of the bulge loop is affected by the type of bulged base, the flanking base pairs [5][6][7][8][9][10], and the loop position in the sequence [11]. Therefore, the ΔG°_{37} of the bulge loop structure needs to be estimated more accurately by modeling these effects in order to design an adequate sequence. We have estimated the ΔG°_{37} of a single bulge loop by using the nearest-neighbor (NN) model. A single bulge loop is apparently the most stable bulge loop structures of any length. If so, a single bulge loop has the greatest possibility of mis-hybridization. We also investigated the effect of loop position on the stability of a single bulge loop for use in sequence design and DNA hybridization simulation.

2 Materials and Methods

Absorbance versus temperature profiles (melting curves) were measured at 260 and 320 nm with a heating rate of 1.0 deg *C*/min on a SHIMADZU UV-1650PC spectrophotometer. The absorbance at 260 nm (ϵ_{260}) was for oligonucleotides, while that at 320 nm (ϵ_{320}) was for the background. We plotted the curves based on the difference ($\epsilon_{260-320}$) between the two absorbancies. The buffer solution was a mixture of 1 M NaCl, 10 mM Na_2HPO_4, and 1 mM Na_2EDTA, with a pH of 7.0. The oligonucleotide concentration (Ct) of each sample was determined based on the absorbance difference using extinction coefficients calculated from dinucleoside monophosphates and nucleotides [2].

In the NN model, the number of thermodynamic parameters for a single bulge is 64. We estimated 40 parameters for a single bulge with a bulged base and no identical flanking base except for the A**A**A-, T**T**T-, C**C**C-, and G**G**G-bulges. These four bulges were estimated and used to investigate the effect of position degeneracy [10] (see Discussion). The thermodynamic parameters for sequences with a single bulge were estimated from the absorbance versus temperature melting curves using plots of the reciprocal melting temperature (Tm^{-1}) vs. lnC_T (called the "vant' Hoff plot") based on the following equation [4].

$$T_M^{-1} = R/\Delta H^{\circ} lnC_T/\alpha + \Delta S^{\circ}/\Delta H^{\circ}. \quad (1)$$

For self-complementary sequences, α in equation 1 was 1, and for non-self-complementary sequences, it was 4.

The slope of the vant' Hoff plot is $R/\Delta H^{\circ}$, and the intercept is $\Delta S^{\circ}/\Delta H^{\circ}$ (see Figure 1). The error in the thermodynamics parameters was estimated based on the linearity of the regression line [3].

The oligomer concentration was varied over a 50-fold range of at least seven data points.

The Tm was estimated as follows. First, the lower and upper base lines were determined based on the regression line in the pre- and post-transition domains, respectively. Next, the Tm was determined based on the point where the melting

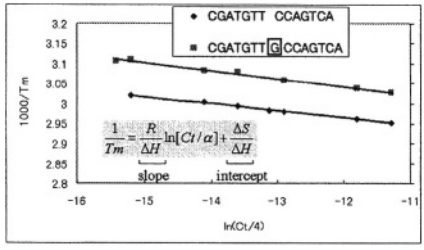


Fig. 1. Vant' Hoff Plot

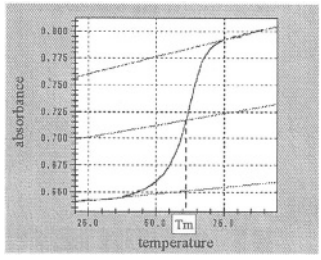


Fig. 2. Estimated Tm

curve intersects the median line between the lower and upper base lines (see Figure 2).

The sequences were designed to minimize the possibility of forming an undesired hairpin or slipped-duplex structure. Further, they were designed to have a single bulge loop in the middle of the sequence in order to have as close to the same effect as that of the loop position as possible (see Discussion).

In the NN model, the bulge loop contribution to duplex stability can be determined based on the thermodynamics difference between a sequence with a single bulge loop and a "core sequence" and then adding back the NN parameter of the flanking base pairs [6].

$$\begin{array}{llll} G\boxed{A}T = CATCCGAG & \boxed{A}TTTACAGC - CATCCGAGTTTACAGC + GT \\ C \quad A \quad\; GTAGGCTC & \quad AAATGTCG \quad\; GTAGGCTCAAATGTCG \quad CA. \end{array} \tag{2}$$

We estimated the NN parameters of the flanking base pairs by using SantaLucia's parameters [4].

A regression line was drawn by using the least-squares method with all data points weighted equally and with the error in C_T negligible compared to that in $1/T_M$ [3].

3 Experimental Results

The plots of T_M^{-1} versus In C_T were linear (coefficient of determination $R^2 \geq 0.96$), indicating that the error in measuring in $1/T_M$ was small.

The thermodynamics parameters of the core sequences and sequences with a single bulge derived from the T_M^{-1} versus In C_T plots and the error obtained based on the linearity of the regression line are listed in Table 1. The contributions of single bulges to the NN thermodynamic parameters are listed in Table 2. In Tables 1 and 2, only the top strand is given, and the boxed bases are bulged bases. For example, 5'-CGCTATGA **A** ACCACTTG-3' in Table 1 shows the structure 5'-CGCTATG A **A** ACCACTTG-3'/3'-GCGATACTTGGTGAAC-5'.

The ΔG°_{37} of a single bulge loop is plotted in Figure 3. The most stable structure was the C **G** T-bulge, while the most unstable one was the C **A** C-bulge.

Table 1. Nearest-Neighbor Thermodynamic Parameters for Core Sequences and Sequences with a Single Bulge. Boxed bases are bulged bases. The error obtained based on the linearity of the regression line

Sequence	ΔH° ($kcal/mol$)	ΔS° (eu)	ΔG°_{37} ($kcal/mol$)
5'-CGCTATGAACCACTTG-3'	-133.64 ± 5.19	-368.3 ± 15.3	-19.40 ± 0.44
5'-CGCTATGA A ACCACTTG-3'	-124.99 ± 5.27	-349.5 ± 15.9	-16.60 ± 0.35
5'-CGCTATGA T ACCACTTG-3'	-122.55 ± 3.42	-343.4 ± 10.3	-16.05 ± 0.22
5'-CGCTATGA C ACCACTTG-3'	-120.89 ± 6.41	-338.6 ± 19.3	-15.86 ± 0.41
5'-CGCTATGA G ACCACTTG-3'	-123.55 ± 7.48	-347.5 ± 22.7	-15.76 ± 0.45
5'-AGCTCGAGATCGATGAGTGT-3'	-167.99 ± 2.08	-456.4 ± 6.0	-26.45 ± 0.22
5'-AGCTCGAGA C TCGATGAGTGT-3'	-167.38 ± 2.18	-463.2 ± 6.4	-23.72 ± 0.20
5'-CACGACTATCTGAACC-3'	-128.37 ± 8.04	-352.3 ± 23.7	-19.10 ± 0.69
5'-CACGACTA G TCTGAACC-3'	-115.11 ± 2.83	-320.3 ± 8.5	-15.77 ± 0.18
5'-GACCTAGACACCATTC-3'	-145.24 ± 7.04	-403.3 ± 20.8	-20.16 ± 0.59
5'-GACCTAGA T CACCATTC-3'	-121.56 ± 4.29	-341.5 ± 13.0	-15.65 ± 0.26
5'-GACCTAGA G CACCATTC-3'	-124.02 ± 2.29	-349.9 ± 7.0	-15.49 ± 0.14
5'-TTAGCAGAGTCCATCG-3'	-120.10 ± 6.39	-326.2 ± 18.8	-18.94 ± 0.57
5'-TTAGCAGA T GTCCATCG-3'	-111.74 ± 3.10	-307.7 ± 9.3	-16.32 ± 0.22
5'-CTACCACTAAGCCTTG-3'	-119.71 ± 4.40	-328.6 ± 13.0	-17.80 ± 0.36
5'-CTACCACTAA C GCCTTG-3'	-113.58 ± 2.26	-317.3 ± 6.8	-15.18 ± 0.14
5'-CTACCACTAAGCCTTG-3'	-119.71 ± 4.40	-328.6 ± 13.0	-17.80 ± 0.36
5'-CTACCACT C AAGCCTTG-3'	-115.36 ± 11.35	-323.3 ± 34.4	-15.10 ± 0.69
5'-CTACCACT G AAGCCTTG-3'	-129.34 ± 1.48	-366.4 ± 4.5	-15.71 ± 0.09
5'-GTACGCCTTGATTCTC-3'	-137.72 ± 4.38	-380.5 ± 12.9	-19.70 ± 0.37
5'-GTACGCCT A TGATTCTC-3'	-115.19 ± 3.78	-322.1 ± 11.4	-15.29 ± 0.23
5'-GTACGCCT T TGATTCTC-3'	-128.85 ± 7.77	-360.7 ± 23.3	-16.98 ± 0.54
5'-GTACGCCT C TGATTCTC-3'	-124.38 ± 4.28	-349.6 ± 12.9	-15.94 ± 0.27
5'-GTACGCCT G TGATTCTC-3'	-124.29 ± 6.73	-349.8 ± 20.4	-15.80 ± 0.41
5'-GTGTATCTGACC-3'	-93.56 ± 1.36	-259.1 ± 4.1	-13.19 ± 0.08
5'-GTGTAT A CTGACC-3'	-85.88 ± 3.64	-245.7 ± 11.5	-9.67 ± 0.08
5'-CGATGTTCCAGTCA-3'	-112.73 ± 1.88	-310.3 ± 5.6	-16.48 ± 0.14
5'-CGATGTT G CCAGTCA-3'	-99.57 ± 4.45	-279.0 ± 13.7	-13.03 ± 0.21
5'-GCTGAATGTCCTAG-3'	-105.46 ± 1.89	-290.3 ± 5.7	-15.43 ± 0.13
5'-GCTGAAT A GTCCTAG-3'	-84.16 ± 6.97	-233.2 ± 21.5	-11.82 ± 0.31
5'-GCTGAAT C GTCCTAG-3'	-99.87 ± 1.82	-280.6 ± 5.6	-12.86 ± 0.08
5'-CACCATTCAGAGCCTA-3'	-135.23 ± 4.36	-370.6 ± 12.8	-20.29 ± 0.39
5'-CACCATTC T AGAGCCTA-3'	-132.93 ± 10.65	-373.0 ± 32.0	-17.24 ± 0.72
5'-CACCATTC G AGAGCCTA-3'	-128.88 ± 4.12	-360.9 ± 12.4	-16.95 ± 0.28
5'-CTACCACTAAGCCTTG-3'	-119.71 ± 4.40	-328.6 ± 13.0	-17.80 ± 0.36
5'-CTACCAC A TAAGCCTTG-3'	-107.48 ± 7.98	-299.3 ± 24.1	-14.67 ± 0.50
5'-CTACCAC G TAAGCCTTG-3'	-142.64 ± 8.40	-404.4 ± 25.3	-17.22 ± 0.55
5'-GTCATATCCAGCGTTC-3'	-152.49 ± 4.79	-424.1 ± 14.1	-20.97 ± 0.41
5'-GTCATATC A CAGCGTTC-3'	-116.72 ± 3.33	-326.7 ± 10.1	-15.40 ± 0.20
5'-GTCATATC T CAGCGTTC-3'	-135.54 ± 4.30	-383.6 ± 13.0	-16.56 ± 0.27
5'-GTCATATC C CAGCGTTC-3'	-123.58 ± 2.23	-344.9 ± 6.7	-16.62 ± 0.15
5'-GTCATATC G CAGCGTTC-3'	-125.42 ± 4.47	-353.3 ± 13.5	-15.84 ± 0.27
5'-CTAAGGTCGTGTATCC-3'	-133.49 ± 6.41	-369.2 ± 19.0	-18.97 ± 0.52
5'-CTAAGGTC A GTGTATCC-3'	-118.07 ± 2.35	-331.6 ± 7.1	-15.22 ± 0.14
5'-CTAAGGTC T GTGTATCC-3'	-119.90 ± 4.34	-337.4 ± 13.2	-15.26 ± 0.25
5'-GGTCAGATACAC-3'	-93.56 ± 1.36	-259.1 ± 4.1	-13.19 ± 0.08
5'-GGTCAG T ATACAC-3'	-90.95 ± 2.32	-260.0 ± 7.3	-10.31 ± 0.06
5'-AGCATTAGACGGACCT-3'	-129.46 ± 8.76	-351.8 ± 25.6	-20.34 ± 0.83
5'-AGCATTAG C ACGGACCT-3'	-116.88 ± 2.71	-323.3 ± 8.1	-16.60 ± 0.19
5'-CATCCGAGTTTACAGC-3'	-129.99 ± 9.36	-356.6 ± 27.5	-19.41 ± 0.82
5'-CATCCGAG A TTTACAGC-3'	-129.41 ± 5.13	-364.2 ± 15.5	-16.46 ± 0.33
5'-CATCCGAG C TTTACAGC-3'	-123.13 ± 5.55	-344.7 ± 16.7	-16.23 ± 0.37

Table 1. continued

5'-CACTAAGGCTGTTCAC-3'	-135.69 ± 5.80	-373.6 ± 17.1	-19.83 ± 0.49
5'-CACTAAGG **A** CTGTTCAC-3'	-121.84 ± 6.51	-342.2 ± 19.7	-15.70 ± 0.41
5'-CACTAAGG **T** CTGTTCAC-3'	-129.26 ± 10.44	-363.9 ± 31.5	-16.39 ± 0.66
5'-CTCAAATGGTACGCAG-3'	-130.66 ± 11.13	-359.7 ± 32.9	-19.09 ± 0.94
5'-CTCAAATG **A** GTACGCAG-3'	-119.73 ± 4.33	-334.1 ± 13.1	-16.12 ± 0.28
5'-CTCAAATG **T** GTACGCAG-3'	-140.13 ± 10.41	-394.6 ± 31.3	-17.73 ± 0.72
5'-CTCAAATG **C** GTACGCAG-3'	-93.75 ± 2.03	-258.3 ± 6.2	-13.63 ± 0.12
5'-CTCAAATG **G** GTACGCAG-3'	-119.98 ± 5.45	-333.5 ± 16.3	-16.54 ± 0.38

Table 2. Contributions of Single Bulges to Nearest-Neighbor Thermodynamic Parameters. Boxed bases are bulged bases

Bulge	ΔH° (kcal/mol)	ΔS° (eu)	ΔG°_{37} (kcal/mol)
A **A** A	0.75 ± 7.40	-3.4 ± 22.1	1.80 ± 0.56
A **T** A	3.19 ± 6.22	2.8 ± 18.5	2.35 ± 0.49
A **C** A	4.85 ± 8.25	7.5 ± 24.7	2.54 ± 0.60
A **G** A	2.19 ± 9.11	-1.4 ± 27.4	2.64 ± 0.63
A **C** T	-6.60 ± 3.09	-27.2 ± 9.1	1.84 ± 0.30
A **G** T	6.06 ± 8.55	11.6 ± 25.3	2.45 ± 0.72
A **T** C	15.27 ± 8.27	39.4 ± 24.6	3.07 ± 0.64
A **G** C	12.82 ± 7.42	31.0 ± 22.0	3.23 ± 0.60
A **T** G	0.56 ± 7.13	-2.5 ± 21.0	1.34 ± 0.61
A **C** G	-1.68 ± 4.99	-9.7 ± 14.9	1.35 ± 0.39
T **C** A	-2.85 ± 12.21	-16.0 ± 36.9	2.13 ± 0.78
T **G** A	-16.83 ± 4.73	-59.1 ± 14.0	1.51 ± 0.38
T **A** T	14.63 ± 5.79	36.2 ± 17.3	3.41 ± 0.44
T **T** T	0.97 ± 8.93	-2.4 ± 26.7	1.72 ± 0.65
T **C** T	5.44 ± 6.13	8.7 ± 18.3	2.76 ± 0.46
T **G** T	5.53 ± 8.04	8.5 ± 24.2	2.90 ± 0.55
T **A** C	-0.52 ± 3.93	-8.8 ± 12.4	2.22 ± 0.11
T **G** C	4.96 ± 4.87	9.1 ± 14.9	2.15 ± 0.26
T **A** G	12.80 ± 7.25	34.3 ± 22.3	2.16 ± 0.35
T **C** G	-2.91 ± 2.70	-13.0 ± 8.2	1.13 ± 0.17
C **T** A	-6.20 ± 11.52	-25.1 ± 34.5	1.60 ± 0.82
C **G** A	-2.15 ± 6.03	-13.0 ± 17.9	1.89 ± 0.48
C **A** T	4.43 ± 9.13	8.3 ± 27.5	1.86 ± 0.62
C **G** T	-30.73 ± 9.51	-96.8 ± 28.6	-0.69 ± 0.66
C **A** C	27.77 ± 5.90	77.5 ± 17.5	3.73 ± 0.46
C **T** C	8.95 ± 6.50	20.5 ± 19.3	2.57 ± 0.49
C **C** C	20.92 ± 5.36	59.3 ± 15.8	2.51 ± 0.43
C **G** C	19.07 ± 6.61	50.9 ± 19.7	3.29 ± 0.49
C **A** G	4.83 ± 6.86	10.4 ± 20.5	1.59 ± 0.54
C **T** G	3.00 ± 7.77	4.6 ± 23.3	1.56 ± 0.58
G **T** A	-5.59 ± 2.76	-23.1 ± 8.6	1.58 ± 0.10
G **C** A	4.38 ± 9.19	6.3 ± 26.9	2.43 ± 0.85
G **A** T	-7.82 ± 10.68	-30.0 ± 31.7	1.50 ± 0.88
G **C** T	-1.54 ± 10.89	-10.5 ± 32.3	1.73 ± 0.90
G **A** C	4.05 ± 8.73	7.0 ± 26.2	1.88 ± 0.64
G **T** C	-3.37 ± 11.95	-14.8 ± 35.9	1.20 ± 0.82
G **A** G	2.93 ± 11.97	5.8 ± 35.4	1.13 ± 0.98
G **T** G	-17.47 ± 15.26	-54.8 ± 45.4	-0.48 ± 1.18
G **C** G	28.92 ± 11.35	81.5 ± 33.5	3.61 ± 0.95
G **G** G	2.68 ± 12.42	6.3 ± 36.7	0.71 ± 1.01

4 Discussion

4.1 Tendency of ΔG°_{37} for Single Bulge Loop

The effect of a bulged base on the ΔG°_{37} is shown in Figure 4. Each bar represents the average ΔG°_{37} of a bulged base (A, T, C, G). Conventional studies produced inconsistent results for the contribution of a bulged base to ΔG°_{37}. LeBlanc et al. concluded that bulged pyrimidines (i.e., T- and C-bulges) are more stable than bulged purines (i.e., A- and G-bulges) [5], while Ke et al.

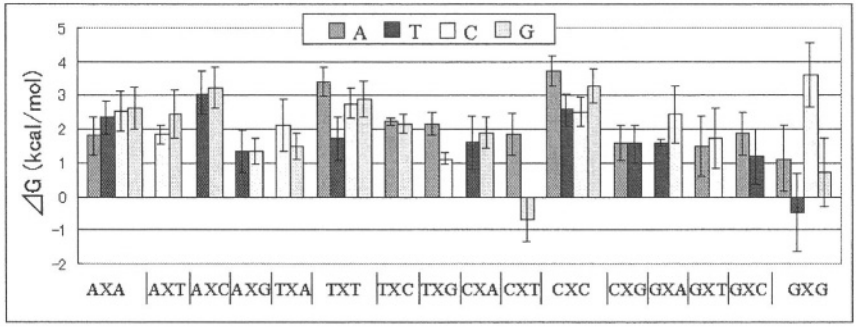


Fig. 3. ΔG°_{37} of Single Bulge

found that bulged purines were more stable than bulged pyrimidines [9]. Our results support neither of these findings (T-bulge (pyrimidine) > G-bulge (purine) $\geq$ A-bulge (purine) $\geq$ C-bulge (pyrimidine)), although the ΔG°_{37} obtained using LeBlanc's results showed excellent agreement with that obtained using our results (see Figure 6). Ke et al. argued that bulged purines are more stable than bulged pyrimidines because purines are generally stacked into a helix, while pyrimidines are extrahelical or intrahelical depending on the sequence context and temperature [9]. However, extrahelical bulges are not always less stable than intrahelical bulges. We concluded that ΔG°_{37} is determined by the stacking energy between the bulged base and the flanking bases (stacked into a helix) or the left flanking base and the right flanking base (extrahelical), rather than the nature of the bulged base (purine vs. pyrimidine).

The effect of the flanking base pairs on ΔG°_{37} is shown in Figure 5. Each bar in Figure 5 represents the average ΔG°_{37} for a flanking base pair (purine$*$purine, purine$*$pyrimidine, pyrimidine$*$purine, pyrimidine$*$ pyrimidine). The ranking of single bulge stability was pyrimidine$*$purine $\geq$ purine$*$purine > purine$*$pyrimidine > pyrimidine$*$pyrimidine. The single bulge loop between two pyrimidines was observed to be less stable than the others. This tendency is consistent with conventional studies [5] [8]. Papanicolaou et al. attributed this phenomenon to the poor stacking of the pyrimidines [8].

4.2 Comparison with Non-NN-model-based Approach

Zhu et al. also studied the ΔG°_{37} for a single bulge loop using an approach different from ours [10]. They modeled ΔG°_{37} based on the stacking energy between the flanking base pairs. They placed the single bulge loops into one of two categories, those with a bulged base and no identical flanking base (Group I) and those with a bulged base and at least one identical flanking base (Group II). They found that Group II bulges tended to be more stable than Group I bulges in the same nearest neighbor environments because of more conformational freedom (they called this effect "position degeneracy"). They derived the following

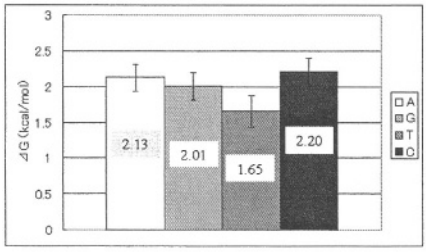


Fig. 4. Effect of Bulged Base

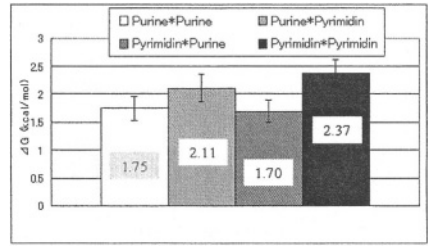


Fig. 5. Effect of Flanking Base Pairs

equation based on their experimental data.

$$\Delta G_{(XNZ)\cdot(X'-Z')} = 2.72 + 0.48\Delta G^{S}_{(XZ)\cdot(X'Z')} - \delta g, \tag{3}$$

where δg is zero for Group I bulges and 0.4 kcal/mol for Group II bulges. The first term, 2.72 [kcal/mol], is the positive free energy due to inserting a base bulge into the duplex. The second term is the free energy contribution of the stacking interaction by the flanking base pairs. The third term represents the effect of position degeneracy. The ΔG°_{37} from our results also reflected the effect of position degeneracy. For example, the A**A**A-bulge was the most stable structure among the A**X**A-bulges (X=A, T, C, or G) (see Figure 3). Zhu's model does not distinguish the type of bulged base or in which sequence (i.e., sense or antisense) a bulged base exists. For example, the stability of 5'-C**X**T-3'/5'-AG-3' (X=A, T, C, or G) equals that of 5'-CT-3'/5'-A**X**G-3' according to Equation 3. However, the ΔG°_{37} derived from our experimental data do not necessarily agree with that from Equation 3. For example, the ΔG°_{37} of the G**T**G-bulge was -0.48 kcal/mol, while that of the G**C**G-bulge was 3.61 kcal/mol. This exception reveals that the bulged base affects the ΔG°_{37} for a single bulge loop. On the other, the ΔG°_{37} of the G**T**G-bulge was -0.48 kcal/mol, while that of the C**T**C-bulge was 2.57 kcal/mol. This exception reveals that the flanking base pairs affect the ΔG°_{37} for a single bulge loop. Therefore, to approximate the ΔG°_{37} for a single bulge loop adequately, we need to consider at least the type of bulged base and its flanking base pairs. Moreover, more detailed experiments are needed to determine whether the NN model is enough to approximate the ΔG°_{37} for a single bulge loop.

4.3 Comparison with Results of Related Work

Figures 6-8 compare the ΔG°_{37} from related work [5][6][7] with that derived from our results. The sequences analyzed are shown in Tables 3-5. Figure 6 shows that the ΔG°_{37} obtained by LeBlanc's results [5] agrees well with that derived from our results. This means that the ΔG°_{37} for single bulge loops can be estimated adequately based on the NN model. The ΔG°_{37} of others [6][7] do not agreement

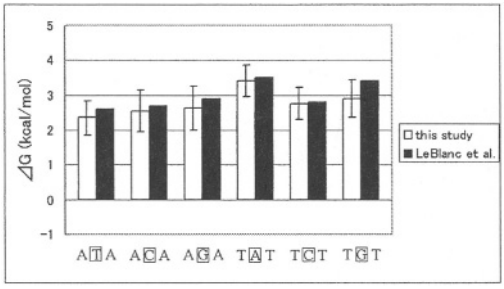


Fig. 6. Comparison between ΔG°_{37} from LeBlanc's Results and That from Ours

Table 3. Sequences analyzed by LeBlanc

5'-GCGAAAAGCG-3'
3'-CGCTTTTCGC-5'
5'-GCGAA T AAGCG-3'
5'-GCGAA C AAGCG-3'
5'-GCGAA G AAGCG-3'
5'-CGCTT A TTCGC-3'
5'-CGCTT C TTCGC-3'
5'-CGCTT G TTCGC-3'

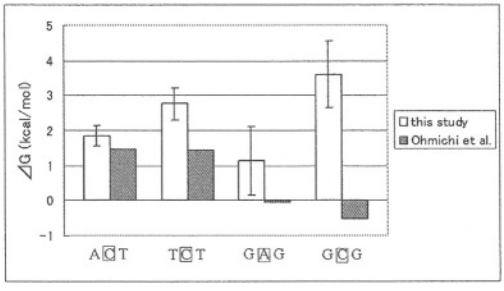


Fig. 7. Comparison between ΔG°_{37} from Ohmichi's Results and That from Ours

Table 4. Sequences analyzed by Ohmichi

5'-TAGGTTATAA-3'
5'-TAGGTTA C TAA-3'
5'-TAGGT C TATAA-3'
5'-TAG A GTTATAA-3'
5'-TAG C GTTATAA-3'

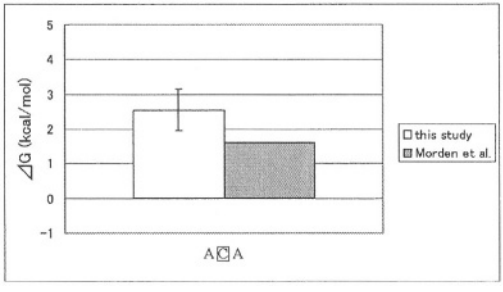


Fig. 8. Comparison between ΔG°_{37} from Morden's Results and That from Ours

Table 5. Sequences analyzed by Morden

5'-CAAAAAAG-3'
5'-CAAA C AAAG-3'

with ours (see Figures 7 and 8). This disagreement may be due to a difference in GC content and the loop position between analyzed sequences, more detailed experiments are needed to clarify this.

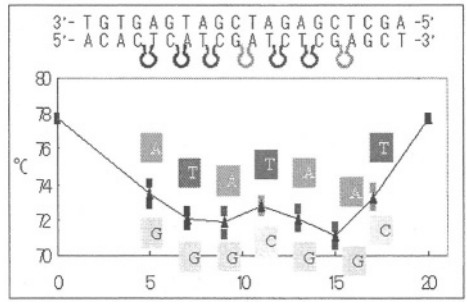


Fig. 9. Effect of Loop Position on Stability. The sequence used here and the position of the loop are shown in the upper part of this graph. The X axis is the position of the loop from the 5'-end of the sequence, which has a bulged base. The Y axis is the Tm. Note that loop positions 0 and 20 are complementary duplexes (not bulge loop structures)

4.4 Effect of Loop Position on Stability of Single Bulge Loop

In the NN model, the contribution of ΔG°_{37} to the bulge loop is assumed to adequately approximated by the effects of the bulged base and the flanking base pairs. However, our previous results revealed that the stability of a single bulge loop is affected by the loop position [11]. Figure 9 shows the relationship between loop position and Tm. Because Tm and ΔG°_{37} are correlated, ΔG°_{37} should be affected by the loop position. The stability (i.e., Tm and probably ΔG°_{37}) of a single bulge loop shows the following tendency. A bulge loop near the end of a sequence is stable, while one near the middle is relatively unstable. One exactly in the middle is stable. This effect should also be modeled in the future.

5 Conclusion

We estimated 40 thermodynamic parameters for DNA duplexes with a single bulge loop in the middle of the sequence based on the NN model. Our results are largely consistent with those of other studies.

We also investigated the effect of loop position on the energy of a single bulge loop. This needs to be quantified by more detailed experiments. We plan to develop an equation for predicting the ΔG°_{37} of a bulge loop structure by using the NN model and loop position effect.

References

[1] James G. Wetmur: Physical Chemistry of Nucleic Acid Hybridization, DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 48, pp. 1-23 (1999)

[2] Donald M. Gray, Su-Hwi Hung, and Kenneth H. Johnson: Absorption and Circular Dichroism Spectroscopy of Nucleic Acid Duplexes and Triplexes, Methods In Enzymology, Vol. 246, pp. 19-34 (1995)

[3] Tianbing Xia, John SantaLucia, Jr., Mark E. Burkard, Ryszard Kierzek, Susan J. Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H. Turner: Thermodynamic Parameters for an Expanded Nearest-Neighbor Model for Formation of RNA Duplexes with Watson-Crick Base Pairs, Biochemistry, Vol. 37, No. 42, pp. 14719-14735 (1998)

[4] Hatim T. Allawi and John SantaLucia, Jr.: Thermodynamics and NMR of Internal G.T Mismatches in DNA, Biochemistry, Vol. 36, pp. 10581-10594 (1997)

[5] Darryl A. LeBlanc and Kathleen M. Morden: Thermodynamic Characterization of Deoxyribooligonucleotide Duplexes Containing Bulges, Biochemistry, Vol. 30, No. 16, pp. 4042-4047 (1991)

[6] Tatsuo Ohmichi, Hiroyuki Nakamuta, Kyohko Yasuda, and Naoki Sugimoto: Kinetic Property of Bulged Helix Formation: Analysis of Kinetic Behavior Using Nearest-Neighbor Parameters, J. Am. Chem. Soc., Vol. 122, No. 46, pp. 11286-11294 (2000)

[7] Kathleen M. Morden, Y. Gloria Chu, Francis H. Martin, and Ignacio Tinoco, Jr.: Unpaired Cytosine in the Deoxyoligonucleotide Duplex $dCA_3CA_3G \cdot dCT_6G$ Is Outside of the Helix, Biochemistry, Vol. 22, pp. 5557-5563 (1983)

[8] Catherine Papanicolaou, Manolo Gouy, and Jacques Ninio: An Energy Model That Predicts the Correct Folding of Both the tRNA and the 5S RNA Molecules, Nucleic Acids Research, Vol. 12, No. 1, pp. 31-44 (1984)

[9] Song-Hua Ke and Roger M. Wartell: Influence of Neighboring Base Pairs on the Stability of Single Base Bulges and Base Pairs in a Fragment, Biochemistry, Vol. 34, No. 14, pp. 4593-4600 (1995)

[10] Jian Zhu and Roger M. Wartell: The Effect of Base Sequence on the Stability of RNA and DNA Single Base Bulges, Biochemistry, Vol. 38, No. 48, pp. 15986-15993 (1999)

[11] Fumiaki Tanaka, Atsushi Kameda, Masahito Yamamoto, and Azuma Ohuchi: The Effect of the Bulge Loop upon the Hybridization Process in DNA Computing, Lecture Notes in Computer Science 2606, Evolvable Systems: From Biology to Hardware, pp. 446-456 (2003)

[12] Udo Feldkamp, Sam Saghafi, Wolfgang Banzhaf, and Hilmar Rauhe: DNASequenceGenerator: A Program for the Construction of DNA Sequences, Preliminary Proc. of Seventh International Meeting on DNA Based Computers, pp. 179-188 (2001)

[13] Satoshi Kobayashi, Tomohiro Kondo, and Masanori Arita: On Template Method for DNA Sequence Design, Preliminary Proc. of Eighth International Meeting on DNA Based Computers, pp. 115-124 (2002)

Mathematical Considerations in the Design of Microreactor-Based DNA Computers

Michael S. Livstone and Laura F. Landweber

Princeton University Department of Ecology and Evolutionary Biology
Guyot Hall, Washington Road, Princeton, NJ 08544 USA
{livstone,lfl}@princeton.edu

Abstract. DNA-based computation in microreactors allows the use of smaller volumes and simplifies automation, reducing both cost and time commitments. We examine ways to construct and implement small microreactor systems implementing analogues of the Boolean functions AND and OR. Relative positions of microreactors (in series and in parallel) are considered, as are different methods of recovery of solution strands, *i.e.* either from the fraction specifically retained in the microreactor (positive selection) or allowed to pass through after non-solution strands are removed (negative selection). Primary consideration is given to the overall accuracy of the system implementing the functions, and a secondary concern for the OR function is a balanced representation of correct solution strands in the final pool. We conclude that positive selection should confer higher accuracy than negative selection and that both AND and OR functions are better implemented by microreactors arranged in series.

1 Introduction

One of the difficulties in the field of DNA computing has been the large amounts of time required to obtain an answer [1-3]. As recently as 2002, the computational steps of a 20-bit, 24-clause 3-SAT problem [2] required 96 hours to complete; this does not include the time required to set up or optimize the experiment or to analyze the results. Two ways to address this problem are automation and miniaturization [4-8]. Many of the rate-limiting steps in previous attempts were those that required manipulation by human beings. For example, a human may be challenged to measure and transfer 1 μl accurately, but a robotic syringe pump could do so accurately, quickly, and repeatedly. Reducing the volume of liquid required to perform an experiment would correspondingly reduce other needs, such as the duration of an electrophoresis step or the amount of energy required to heat the solution.

It has therefore been proposed that microfluidic systems might be an appropriate way to automate DNA computing [4,6,7,9]. Such a system consists of multiple small chambers, each with internal volumes on the nanoliter scale, connected by channels and filled with liquid which flows through under the control of an external pump. The chambers, or "microreactors," may also contain functionalized beads, so individual biochemical reactions may be performed in each.

Microreactors can be arranged to implement the logical functions AND and OR, thereby mimicking the logic gates used in conventional computers. This mimicry,

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 180-189, 2004.

however, takes the form of a sorting function, in which DNA strands representing correct solutions to a Boolean expression are separated from incorrect solutions, and not the logical processing of a standard electronic gate. Therefore, the effectiveness of such a system is limited by the ability of individual microreactors to separate correct from incorrect strands. This paper analyzes the properties and effectiveness of these microreactor-based AND and OR functions constructed in several different configurations.

2 Definitions

The following conventions, terms, and concepts will be useful for this analysis.

Boolean literals are printed in boldface italic type. Capital letters indicate the TRUE state, and lowercase letters the FALSE state.

A DNA *bit* is a short (~15nt) sequence that has been designated to represent a Boolean literal, *e.g.* ***A*** or ***a***. Bits may be concatenated or otherwise combined to represent the TRUE/FALSE states of multiple literals, *e.g.* ***AB, Ab,*** etc.

An individual *microreactor* (or *reactor*) consists of a chamber with an input and an output channel. The reactor bears immobilized oligonucleotides complementary to one bit. By convention, the flow of liquid in all of the reactor diagrams shown below proceeds left-to-right.

As a library of DNA strands flows through the microreactor, those containing the bit complementary to the immobilized sequence should hybridize and be removed from the solution. Recovery and purification of the captured strands constitutes *positive selection*; removal of non-solution strands by binding to reactors and recovery of the solution strands from the flow-through constitutes *negative selection.*

Since not all strands containing bits complementary to the immobilized oligonucleotides will be retained, it is useful to describe imperfect binding and its consequences. *Efficiency* (ε) is the fraction of strands applied to a microreactor that should bind and actually do. The fraction of *rogue molecules* (ρ) are those which should bind but don't. $\varepsilon+\rho=1$. The *accuracy* of a protocol or microreactor network is the fraction of strands recovered at the end of the experiment that represent correct solutions to the question asked.

Some Boolean expressions have multiple solutions. For example, *"**A*** OR ***B**"* is satisfied by ***AB, Ab,*** and ***aB,*** but not by ***ab***. The first three combinations are equally valid solutions. *Bias* describes the degree to which equivalent solutions are over- or underrepresented in the final pool of recovered strands. For the small number of applicable cases described below, this qualitative description of bias will suffice.

Microreactors arranged in *parallel* or in *series* resemble electronic circuits with the same arrangement, as shown in the diagrams below.

As mentioned above, microreactor circuit elements do not perform the same functions as electronic logic gates. For example, an AND gate takes two inputs and returns TRUE if they are both true. In contrast, a microreactor system implementing ***A*** AND ***B*** takes a mixture of DNA strands and allows the user to recover only those where ***A*** and ***B*** are both true. Therefore, microreactors act as *sorters* rather than as gates.

3 Assumptions

We make several assumptions to simplify the analysis.

We assume that ε, the fraction of strands complementary to the immobilized sequences and which actually bind, is the same for all reactors. In reality, ε is a function of the concentrations of free and immobilized DNA, flow rates, reactor volumes and shapes, buffer composition and temperature, length and sequence of the relevant bit, the rate constant and free energy of hybridization, and other factors affecting hybridization kinetics. These factors are summarized in the single variable ε, which is assumed to be constant for all microreactors. While this cannot hold for entire systems of reactors, it is a reasonable approximation for small systems like the two-reactor sorters described below. More precisely, we assume that the experiment is performed under conditions where ε is constant within small regions of a larger system.

We also assume that no DNA strands are lost due to nonspecific binding to any of the components of the system, such as the walls of reactors. In reality, some materials common in the manufacture of microreactors adsorb DNA. While this may be a confounding factor leading to loss of yield at each step, we assume here for the purpose of simplicity that any components that come into contact with DNA in solution can be made of inert material, can be treated in such a way as to block nonspecific binding, or affect the solution uniformly and can therefore be assumed to be minimal until a more detailed analysis is performed.

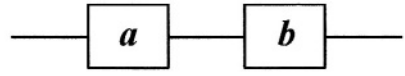
4 Structures of AND and OR Sorters

This section analyzes how well the expressions *"**A*** AND ***B**"* and *"**A*** OR ***B**"* can be implemented by microreactor systems. Both positive and negative selection, as well as parallel and series arrangements of microreactors, are considered, and the relative merits of each are examined. At first glance, it may seem that AND sorters should be arranged in series and OR sorters in parallel, but we will consider the opposite arrangements as well.

4.1 AND Sorters Arranged in Series Performing Negative Selection

Recall that, under negative selection, the correct solutions are collected from the flow-through; therefore, such a reactor is built by concatenating two microreactors, one bearing the DNA complement of ***a***, and the other the complement of ***b***. What should happen to each of the four possible solutions–***AB***, ***Ab***, ***aB***, and ***ab*** (first column, below)–as they pass through the system? If one unit of ***AB*** is applied to the system (first row, second column), none of it should bind specifically to the ***a*** reactor (third column), and all of it should pass through (fourth column); also none should bind to the ***b*** reactor (fifth column), so all of these strands should pass through and be collected (sixth column). When one unit of ***Ab*** is applied (second row), all of it should pass through the ***a*** reactor, but a fraction (ε) should bind to the ***b*** reactor, allowing some (ρ) to pass through and be collected. For ***aB*** (third row) ε should bind to the ***a***

reactor, and the remaining ρ should pass through the ***b*** reactor to the end. For ***ab*** (fourth row), ε should bind to the first reactor, and ε of the remainder should bind to the second reactor, allowing ρ^2 through to the end.



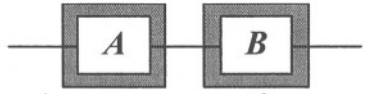
AB	1	0	1	0	1
Ab	1	0	1	ε	ρ
aB	1	ε	ρ	0	ρ
ab	1	ε	ρ	ερ	ρ^2

The accuracy is the amount of the output that represents ***AB,*** the correct solution to *"**A*** AND ***B**,"* divided by the total output of the system, which is the sum of the values in the rightmost column. That is:

$$Accuracy = \frac{1}{1+2\rho+\rho^2} = \frac{1}{(1+\rho)^2} = \frac{1}{(2-\varepsilon)^2}$$

In other words, for an optimistic binding efficiency of 90%, the accuracy of the output of a single AND sorter would be 83%. For a more realistic binding efficiency of 50%, the overall accuracy would be 44%.

4.2 AND Sorter Arranged in Series Performing Positive Selection



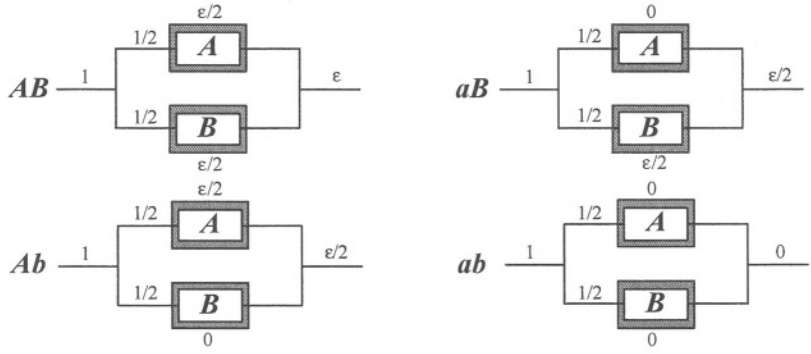
AB	1	ε	ε^2
Ab	1	ε	0
aB	1	0	0
ab	1	0	0

In positive selection, it is necessary to add a design element in order to recover captured strands. For example, heating elements (gray) attached to the reactors can be turned on to melt the annealed strands or switched off to permit annealing. To operate this sorter, first turn the heat off in chamber ***A*** and on in chamber ***B***, then apply one unit each of DNA strands. A fraction (ε) of ***AB*** and ***Ab*** should be retained in chamber ***A*** (third column), while ***aB*** and ***ab*** are not. Everything not retained should flow through, with nothing binding in chamber ***B*** since the heat is on. Next, wash the unbound molecules through, stop the flow, turn the heat on in chamber ***A*** and off in chamber ***B***, allow the reactors to reach the appropriate temperature, then re-start the flow. The molecules bound in chamber ***A*** should melt off and flow into chamber ***B***. Of the ε of ***AB*** retained in chamber ***A***, ε should also be retained in chamber ***B*** (assuming the solution has cooled sufficiently), for an overall retention of ε^2 (fourth column). None of the ***Ab*** released from reactor ***A*** should bind in chamber ***B*** (fourth column), since ***Ab*** fails ***B***. No ***aB*** or ***ab*** should remain, since they should have been

washed out. Finally, wash out the unbound molecules, stop the flow, turn the heat back on in chamber ***B***, re-start the flow, and collect the molecules eluted from chamber ***B***.

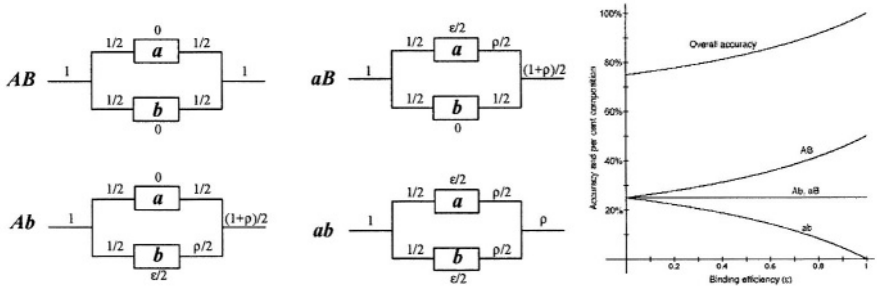
Of the one unit of ***AB*** molecules applied to this system, ε^2 should be recovered (fourth column), and neither ***Ab, aB,*** nor ***ab*** should be recovered. Therefore, the accuracy of an AND sorter arranged in series and performing positive selection is theoretically 100%, while the upper bound for accuracy of such a sorter performing negative selection is less than 100% (see case 1).

4.3 OR Sorter Using Positive Selection and Arranged in Parallel



When reactors are arranged in parallel, there is a branch point at which half the strands are directed in each direction. If one unit *of* ***AB*** is applied, one-half unit will pass through each of the branches. With a binding efficiency of ε, ε/2 should bind to each branch of the sorter and be recovered after elution by heating, for a combined total of ε (see figure). For ***Ab*** and ***aB,*** ε/2 should bind in one chamber and nothing in the other. ***ab*** should not bind. Therefore, the overall accuracy of this configuration is theoretically 100%. However, bias is a serious consideration, since the three correct solutions are disproportionately represented in the final pool in the ratio 1:1:2.

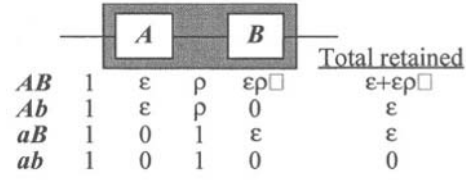
4.4 OR Sorter Using Negative Selection and Arranged in Parallel



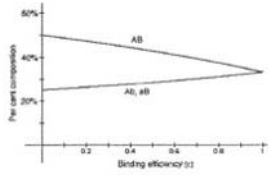
As with other negative selection protocols, some of the ***ab*** strands, which should be removed, are not. Interestingly, ***Ab*** and ***aB*** each constitute 25% of the answer pool, no matter what the binding efficiency is, so ***AB*** and ***ab*** combine to comprise the

remaining 50% of the answer pool. Therefore, as the binding efficiency increases, so does the overall accuracy of the system, as well as the bias. At ε=100%, the output exists in the same ratios as in a parallel OR sorter using positive selection. Therefore such a sorter can perform no better than if it were using positive selection.

4.5 OR Sorter Using Positive Selection and Arranged in Series



					Total retained
AB	1	ε	ρ	ερ□	ε+ερ□
Ab	1	ε	ρ	0	ε
aB	1	0	1	ε	ε
ab	1	0	1	0	0



It is possible to alter the configuration from part 2 to be an OR sorter simply by merging the two heating elements into one and adjusting the protocol. With the heat off, apply the mixture of DNA strands to the system and wash off the unbound fraction. Then turn on the heat and elute whatever binds either to reactor ***A*** or reactor ***B***. For ***AB,*** ε should be retained in reactor ***A,*** ρ should pass through reactor ***A,*** and ερ should be retained in reactor ***B,*** for a total of ε+ερ retained (first row). As with other positive selection protocols, accuracy is theoretically 100%. As the binding efficiency increases, ***AB*** constitutes less of the final pool, and ***Ab*** and ***aB*** constitute more, until ε=100% and each constitutes one-third of the final pool. Even at ε=50%, ***Ab*** and ***aB*** each comprise 29% of the final pool, and ***AB*** 43%. For all values of ε, bias is lower for this configuration than for an OR sorter using positive selection and arranged in parallel.

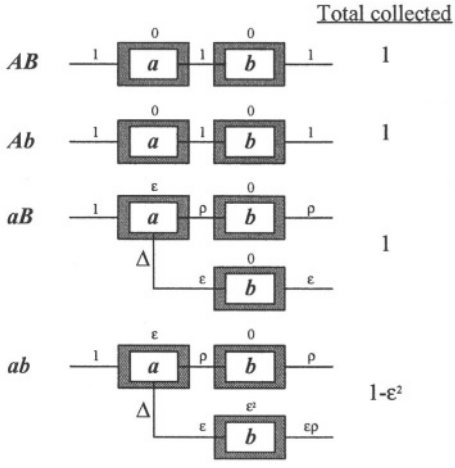
4.6 OR Sorter Using Negative Selection and Arranged in Series

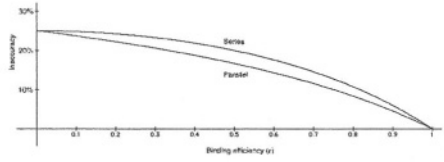
This protocol requires collection of strands from the flow-through in two separate steps. Starting with heater A off and heater B on, apply the mixture of DNA strands to the system, ***ab*** and ***aB*** should stick in reactor ***a***; ***AB*** and ***Ab*** should flow through and be collected. Then, turn the flow off, heater A on, and heater B off, and allow the system to come to temperature. The strands should melt off reactor ***a***. Turn the flow back on. ***ab*** and ***aB*** should leave reactor ***a,*** ***ab*** should be retained in reactor ***b,*** and ***aB*** should flow through and be collected.

Under this protocol, all of the ***AB, Ab,*** and ***aB*** strands should be recovered, and $1-\varepsilon^2$ of the ***ab*** strands should pass through unperturbed. Therefore, there is no bias among the correct solutions, but the accuracy of the protocol is less than 100%. In contrast, an OR sorter using positive selection and arranged in series has 100% accuracy, but imperfect bias (see case 5). Since bias is a secondary consideration to accuracy, positive selection works better in this case.

Also, among OR sorters using negative selection (compare case 4), sorters arranged in series cause a higher percentage of the final pool to consist of incorrect ***ab*** strands (see graph). For all values of ε, the *inaccuracy* of the sorter, *e.g.* the fraction of the final pool that consists of ***ab*** strands, is higher for sorters arranged in series than

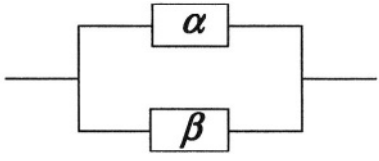
those in parallel. Therefore, if your application requires an OR sorter utilizing negative selection, it should be arranged in parallel, not in series.





4.7 AND Sorters Arranged in Parallel, Positive or Negative Selection

Parallel AND sorters cannot be built. For proof, consider the following configuration:



Here, α and β represent either ***A*** and ***B*** or ***a*** and ***b***. Consider the fraction of the solution that flows through the α branch. Whatever strands bind to the reactor will be either ***A*** or ***a***, but will be an equimolar mixture of ***B*** and ***b;*** the same is true for the strands that flow through. Therefore, there is no population of strands that can be set to ***AB,*** and the sorter cannot implement the AND function.

5 Discussion

Of the eight possible ways to construct AND and OR sorters, six can implement the desired function. Those utilizing negative selection allow a fraction of the strands representing incorrect solutions to flow through into the final pool, causing the accuracy of the device to be less than 100%. In contrast, positive selection does not force the accuracy of the device to be imperfect, but may cause the yield of strands representing correct solutions to be low. In a system representing a Boolean problem with multiple clauses and operating under positive selection, the maximum amount of DNA recovered at the end is the amount that is retained in the reactors representing the first clause, and it diminishes at every AND statement. Difficulties with yield could possibly be overcome by optimizing the system to increase the value of ε or by introducing a PCR step at an intermediate stage. The latter would have to be designed

judiciously in order to avoid introducing errors (*i.e.* mutations), and perhaps should include a subsequent step in which mutated molecules are removed from the system. Accordingly, the trade-off between accuracy and yield is not entirely symmetrical, since low yields are not as significant a problem as low accuracy.

Because negative selection is theoretically simple to implement–just apply DNA to the input and collect the answer from the output–it is tempting to ascribe to it certain favorable properties. For example: Negative selection may obviate the need for PCR amplification of the solution pool, and the errors inherent to PCR, because correct solutions pass through the reactor systems unperturbed and therefore in high yield. However, consider case 1 above, an AND sorter. Even with ε=90%, overall accuracy is predicted to be 83%. Which is greater: the degree of inaccuracy that would arise in this sorter, or that which would arise due to PCR?

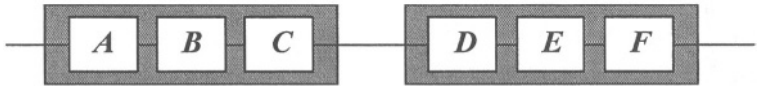
An OR sorter utilizing positive selection and arranged in series performs better, *i.e.* it has the same accuracy and lower bias, than if it were arranged in parallel. It may seem counterintuitive that OR can be implemented in series, but actually such a configuration better portrays the logical configuration of the OR function. To wit, those strands which satisfy ***A*** bind in the first chamber and need not be examined in the second, while those that fail ***A*** get a second chance to bind in the second chamber; ***AB*** is not overrepresented because it binds in *both* chambers, but rather because it binds in *either* chamber.

Furthermore, this configuration is indistinguishable from the case where both ***A*** and ***B*** beads are mixed together in a single microreactor, a setup analogous to that used by Braich *et al.*[2] Here, each clause of a 3-SAT problem was represented as a cylindrical glass module containing DNA strands complementary to the relevant bits immobilized to a polyacrylamide matrix, and strands were released from one module and passed to the next by heating. Microfluidics may offer some advantages in flexibility over (micro)electrophoresis. For example, it is probably easier to make a variety of functionalized beads and arbitrarily load them into reactors than to cast many functionalized polyacrylamide gels. It is, however, not recommended that both technologies be used in the same system, since a gel would prevent liquid from flowing through a fluidic system.

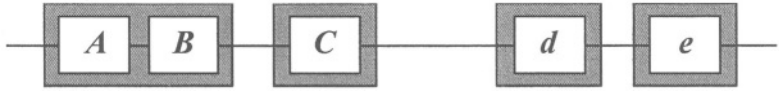
While AND and OR sorters mimic the behavior of AND and OR gates, there is still no simple and obvious way to implement the microreactor analogue of an inverter, *i.e.* a NOT gate. An AND gate takes two bits as input and outputs TRUE if they are both TRUE, and FALSE otherwise. An AND sorter takes as input a mixture of molecules and keeps only those where both bits represent TRUE. Therefore, an AND sorter keeps the molecules that would produce TRUE on an AND gate and fails those that would produce FALSE; similar behavior is seen in OR sorters. However, a NOT sorter must behave differently from a NOT gate. A NOT gate transforms TRUE into FALSE, or vice versa, but a microreactor cannot receive strands representing ***A*** and transform them into ***a***. Rather, a NOT sorter receives a mixture of strands representing ***A*** or ***a*** and keeps those representing ***a***. This behavior is sufficiently different from a NOT gate that a NOT sorter cannot properly be deemed its analogue.

All logical circuits can be built from a set of AND, OR, and NOT gates, but, in the absence of a proper NOT sorter, it may not be possible to build microreactors implementing all Boolean expressions. However, many types of problems, including 3-SAT, can be implemented with only AND and OR sorters.

Since both AND and OR sorters have maximum efficacy when arranged as microreactors in series and utilizing positive selection, a microreactor-based DNA computer could simply be the direct physical manifestation of the Boolean expression it represents: a long string of reactors each representing a literal, connected by heaters implementing the AND and OR functions, and divided into groups each representing a single Boolean clause. Therefore, a microreactor system implementing the two-clause 3-SAT problem *"(**A** OR **B** OR **C**) AND (**D** OR **E** OR **F**)"* may look like this:



More complicated Boolean expressions such as *"((**A** OR **B**)* AND *(**C**)* OR (NOT *(**D** OR **E**)*)" can also be solved. In this case, since there is no sorter equivalent of a NOT gate, it is necessary to change the expression "NOT *(**D** OR **E**)*" to its equivalent *"**d** AND **e**."* Then, the following system can be used:



To operate this system, start with the heat on in chambers ***C*** and ***e*** and off in the others, then apply the mixture of strands. Whatever strands satisfy *"**A** OR **B**"* should be retained in one of the first two reactors, and the rest should flow through reactors ***A*** and ***B***, and also ***C*** since it is hot. (For the purpose of this discussion, it is simplest to assume that $\varepsilon=1$.) Of these, those that satisfy ***d*** should be retained in chamber ***d***, and the rest should flow out since chamber ***e*** is hot. Turn the heat off in chamber ***C*** and on in chambers ***A*** and ***B***. Those strands bound in ***A*** and ***B*** should be released, and those that satisfy ***C*** should be retained in chamber ***C***, with the remainder flowing through and either being retained in chamber ***d*** or flushed through the system. At this point, those strands satisfying the first part of the expression, *"(**A** OR **B**)* AND *(**C**)*," should be bound in chamber ***C***, and the rest should have flowed into chamber ***d***, either being retained (if they satisfied ***d***) or flushed out. Next, turn the heat off in chamber ***e*** and on in ***d*** to implement the "***d*** AND ***e***" function (case 2, above). Finally, turn the heat on in chambers ***C*** and ***e*** to elute the solution strands.

Many Boolean expressions can be constructed using only AND and OR. Under certain circumstances, it is possible to replace the NOT function with an algebraic equivalent. Therefore, microreactor-based DNA computers, while possibly unable to solve all Boolean expressions, are theoretically able to solve many.

6 Conclusion

Microreactor systems potentially represent a simple, powerful, and inexpensive way to implement DNA computers capable of solving Boolean logic problems. Positive selection allows theoretically perfect accuracy, but the maximum theoretical accuracy possible under negative selection is less than 100%. Potential difficulties due to low yield in positive selection may be overcome without sacrificing accuracy. Both AND and OR functions can be built by concatenating multiple reactors. While many Boolean expressions can be built, and therefore solved, using AND and OR functions,

restrictions on the ability to build a NOT function using microreactors establishes a barrier to solving every Boolean expression. This is an example of the belief that DNA-based computers ultimately may not compete directly with electronic computers, but may find a niche of their own.

Acknowledgements

The authors wish to thank Danny van Noort and Ron Weiss for discussion and comments. This work was supported by NSF awards 9875184 and 0121405 and DARPA award F30602-01-2-0560.

References

[1] Adleman, L.M. (1994) Molecular computation of solutions to combinatorial problems. *Science* 266 (5187), 1021-1024.

[2] Braich, R.S. *et al.* (2002) Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* 296 (5567), 499-502.

[3] Faulhammer, D. *et al.* (2000) Molecular computation: RNA solutions to chess problems. *Proc Natl Acad Sci USA* 97 (4), 1385-1389.

[4] van Noort, D. *et al.* (2002) DNA Computing in Microreactors. In *7th International Workshop on DNA-Based Computers, DNA7* (Jonoska, N. and Seeman, N., eds.), pp. 33-45, Springer, Berlin ; New York.

[5] Suyama, A. (2002) Programmable DNA computer with application to mathematical and biological problems. In *Eighth International Meeting on DNA-Based Computers (Preliminary Proceedings),* pp. 91 (see also p. 79, 331, and 332)

[6] McCaskill, J.S. (2001) Optically programming DNA computing in microflow reactors. *Biosystems* 59 (2), 125-138.

[7] Gehani, A. and Reif, J. (1999) Micro flow bio-molecular computation. *Biosystems* 52, 197-216

[8] Livstone, M.S. *et al.* (2003) Molecular computing revisited: a Moore's Law? *TRENDS in Biotechnology (in press)* not known

[9] Chiu, D.T. *et al.* (2001) Using three-dimensional microfluidic networks for solving computationally hard problems. *Proc Natl Acad Sci USA* 98 (6), 2961-2966.

Towards a Re-programmable DNA Computer

Danny van Noort and Laura F. Landweber

Department of Ecology and Evolutionary Biology, Princeton University, NJ, USA
{danny, lfl}@princeton.edu

Abstract. Microreactors lend themselves to a relatively simple implementation of DNA computing. Not only is the design of the DNA library critical for the success of the system but also the architecture of the microfluidic structure. Microreactors can be configured as Boolean operators. This paper will show that biomolecular computing can be performed with elementary building blocks, analogous to electronic logic gates. These logical operations will be performed using negative selection. Furthermore, an alternative bead barrier is introduced which can render the computer re-programmable.

1 Introduction

Biomolecular computing involves a multi-disciplinary approach consisting of molecular biology, microsystems technology, sensing and information technologies. There has been intensive research on the possibility of using biological macromolecules such as DNA to perform calculations, thus simulating the digital information processing procedures performed by conventional computers. The largest problem solved to date is a 20-variable instance of a SAT problem [1] using electrophoresis and gel-filled glass modules. Alternative approaches incorporated, among others, RNA [2], self-assembly [3] and pipetting robots [4].

Microflow technology will prove to be an important tool to realise molecular computing [5, 6] with DNA, RNA or proteins. The advantages with this technology are, for example, the use of small volumes of biological solutions in the nanoliter range and increased reaction rates due to reduced diffusion time. The microflow structures provide control over the flow of solutions, *i.e.* the flow of information. Furthermore, microflow structures can be designed to be problem-specific or re-configurable [7, 8].

Computational problems can in principle be solved on a cascade of these microflow reactors in a network, which executes a series of logic operations.

2 The Microreactor

2.1 Reactor Design

Selection of single stranded DNA molecules (ssDNA) was performed in microreactors with a volume of less than 2 nl. To enhance the selection performance, the reactors were filled with beads (5.6 μm in diameter, Bangs Laboratories Inc., IN)

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 190-196, 2004.

to increase the surface area approximately 10-fold. The flow channels with a width of 25 μm have a different depth (3 μm) from that of the reactor (18 μm), effectively functioning as bead barriers. To deliver the beads to the reactor, a channel is connected to the reactor with the same depth and a width of 50 μm and is closed after the beads have been delivered. To optimize the flow pattern, the microreactors were designed ellipsoidal (Fig. 1), such that the flow followed the contour of the reactor.

2.2 Fabrication Step

The microfluidic structures, including the flow channels, microreactors and bead delivery channels, were made in PDMS (Polydimethylsiloxane ,Sylgard 184, Dow-Corning, MI). The following technological steps were performed.

A multi-depth master was fabricated by photo-lithographic patterning of a negative photoresist (SU 8-2002 and 2007, Microlithography Chemical Corp. Newton, MA) using two different photo-masks of high resolution printed foils (Pageworks, MA). A specified amount of PDMS was then cast against the master. Curing the polymer and releasing it from the master yielded a replica containing the required structures. Inlets and outlets were made by punching holes through the PDMS. The structures were sealed irreversibly by oxidising the PDMS mould and a 0.17 mm microscope coverslip (Fisher Scientific, PA), after which they were brought into contact. Tubes were inserted into the inlets and connected to a syringe pump (Harvard Apparatus, MA). To measure the hybridisation dynamics, the microfluidic system was placed under a fluorescence microscope (Nikon USA).

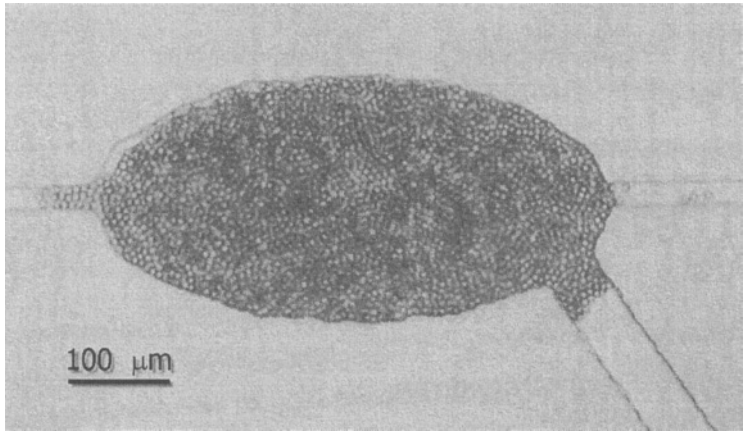


Fig. 1. Ellipsoidal microreactor within a 200x450 μm rectangle filled with 5.6 μm beads. From left to right is a 25 μm flow channel, while in entering from the bottom, right hand corner is the 50 μm wide bead delivery channel

2.3 Logic Gates

Logic problems can be implemented with microflow reactors. Hybridisation between two complementary ssDNA strands is a selection, *i.e.* a YES or NO. AND-operations are performed by using two microflow reactors in series, while OR-operations can be

done with two microflow reactors in parallel (Fig. 2). With these operations it is possible to create simple Boolean statements.

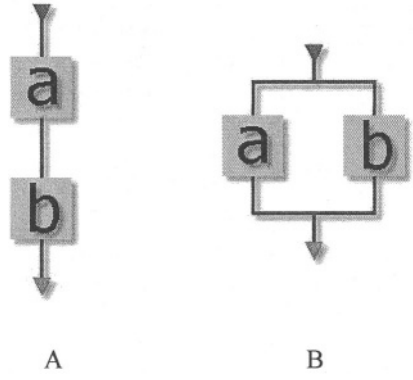


Fig. 2. (a) AND operation; (b) OR operation

3 Negative Selection

A designed DNA strand contains bit information in the sequence of its nucleotides. Single-stranded DNA molecules can be extracted, from a sequence space $\{S_i\}$ (the DNA-library), by using complementary capture probes (CP), a short-single DNA strand. Negative selection discards S_k from the sequence space $\{S_i\}$, while the rest of the sequences $(S_{1,\ldots,k-1,\,k+1,\ldots,N})$ flow to the next microflow reactor, *i.e.* selection.

In our case, a negative selection is performed in microflow reactors in which the CPs are immobilised to the surface of beads. Biotinylated ssDNA molecules are immobilised to streptavidin functionalised beads with a diameter of 5.6 μm.

The main reason for using negative selection is the simplicity of the system design, fabrication and operation. Another reason is that all strands representing correct answers should pass through the system with minimal manipulation (although some non-specific binding will probably occur on the way), thereby minimizing loss of desired strands. Further research will be needed to estimate the error rate. This method is comparable to negative affinity columns used in chromatography.

Initial data shows that negative selection is feasible. At first measurements were performed with different concentrations (from μM to fM) of $d(A)_{25}$ (from Integrated DNA Technologies Inc., IA) to determine the maximal hybridisation to the immobilised biotinylated $d(T)_{25}$. Fluorescence measurements with the intercalater YOYO-1 (Molecular Probes Inc., OR) showed that at a concentration of 0.5 μM (10 μl of d(A)25 with a flowrate of 1 μl/min) a maximal intensity (in arbitrary units) of $6.2\ 10^4$ was detected, while at 0.5 fM this was $5.5\ 10^3$, which is 11-fold smaller. Looking at the latter case, there was more than a 2-fold drop in hybridization capacity to the beads over the length of the reactor (see Fig. 3). This means that there is a sufficient drop in the concentration of the ssDNA to cause a drop in hybridisation dynamics. To check the above conclusion, a long meander was fabricated to monitor this effect of hybridization reduction over distance. Initially 10 μl of 0.5 μM d(A)25 solution was injected. Then under the same conditions a run was performed in a 50

μm wide channel filled over a length of 20 mm with the same 5.6 μm beads as were used in the microreactors. The result is shown in Fig. 4. By comparing the maximal intensities of hybridization, as measured above for different concentrations, it can be seen that towards 2/3 of the channel the concentration has dropped from μM to fM scale, 9 orders of magnitude smaller. This confirms that negative selection in microsystems is in principle possible.

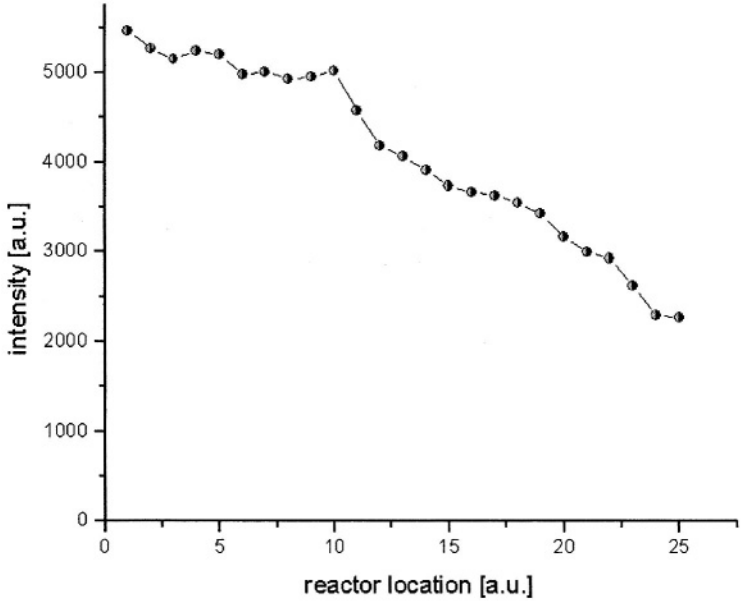


Fig. 3. The maximal intensity of hybridization of a 0.5 fM solution versus the location in the microreactor, 0 being the input

4 Bit Manipulation

The problem of having a DNA-library $\{S_i\}$, with all the possible solutions encoded, is its size. This is recognized as one of the major problems in DNA computing. One way to circumvent this problem is to manipulate single bit representations. In previous research on DNA computing in microreactors [6-8] it was suggested to combine the output flows of the OR statement before proceeding to the next logic operation. This set-up only allows for manipulations of DNA-words. When performing single bit manipulations and combining the flow paths after the selectors in the same manner, it is not possible to distinguish which bit passed which reactor. Take for example a 2-bit word representation $\{0_A0_B, 1_A0_B, 0_A1_B, 1_A1_B\}$, where 0_A and 1_A are the 1st bit (A-bits) with value 0 resp. 1, 0_B and 1_B are the 2nd bit (B-bits) with value 0 resp. 1. $AvB=1$ would have the solution $\{1_A0_B, 0_A1_B, 1_A1_B\}$ (Fig. 5a). However, when just using single bit representation $\{0_A, 1_A, 0_B, 1_B\}$ the solution after the A-reactor would be $\{1_A, 0_B, 1_B\}$ while after the B-reactor this would be $\{0_A, 1_A, 1_B\}$ (Fig. 5b). If the reactor configuration would be the one as shown in Fig. 5a, the outputs of these selections would be identical to the bit set as before the selection,

albeit in a different ratio. By keeping the flows separated (as shown in Fig. 5b) the selected bits can continue to the next stage of selections and single bit manipulation can be performed again in the same manner.

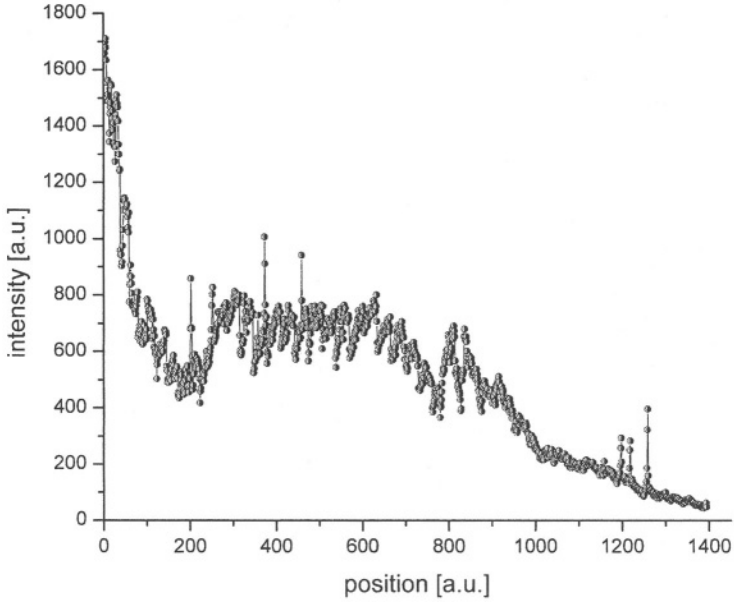


Fig. 4. Intensity profile of hybridization of DNA with an initial concentration of 0.5 μM ssDNA injected versus the location in the channel. After 2/3 of the channel, the concentration has dropped to an order of fM

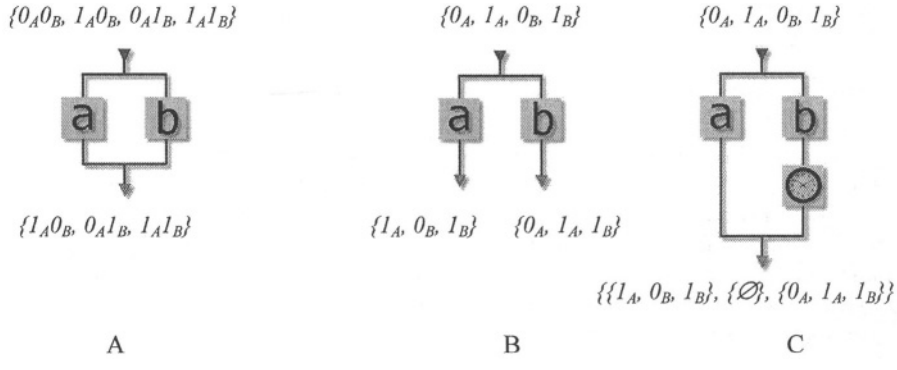


Fig. 5. Two configurations of OR operators. (a) can only be used with a full set of solution strands $\{S_i\}$, while (b) can be used to sort single bits. (c) an OR operator with a delay after a selector. {Ø} depicts the carrier buffer separating the bit information

There is, however, a drawback. The number of selectors will double after each clause in this case, independent of the number of bits. In the worst case, for example, to solve a *n* clause 3-SAT problem, 3^n selectors would be required. This problem can

be overcome by using a delay after, say, the right selector and then recombining the flows (Fig. 5c), reducing the number of selectors to *3n,* for the above mentioned example. A signal similar to electronic handshakes will emerge with the peaks being the bits and the valleys the carrier solution. To avoid diffusion of the bits, however, the carrier solution should be of hydrophobic nature, while the bits are solved in their native buffer.

5 Alternative Bead Barriers

A more flexible method to trap beads is by using valves in a certain location in a fluidic network. These valves were fabricated on top of the flow channels and were made of PDMS as well [9]. A thin layer of PDMS (~40 μm), which was spin coated, acts like a membrane between the pneumatic actuator and the flow channel. When pressure is applied, the membrane is pushed into the underlying channel, effectively blocking the flow. The degree to which the channel is closed depends on the pressure applied, making this valve comparable to potentiometer in electronics. By allowing the valve to close partially, beads were captured while the flow could continue, turning this device into a bead barrier (see Fig. 6).

One of the advantages is that the beads can be loaded through the same channel structure as the flow, so no alternative bead delivery channels are needed, while the valves are used to control the distribution of beads. Furthermore, the valve-system is a completely different circuit from the flow structure, which means there will be no interference with the flow due to the structure, as is the case with bead delivery channels. After the computation has been performed, the beads can be flushed out by opening all the valves, thus making the structure reusable and therefore reprogrammable.

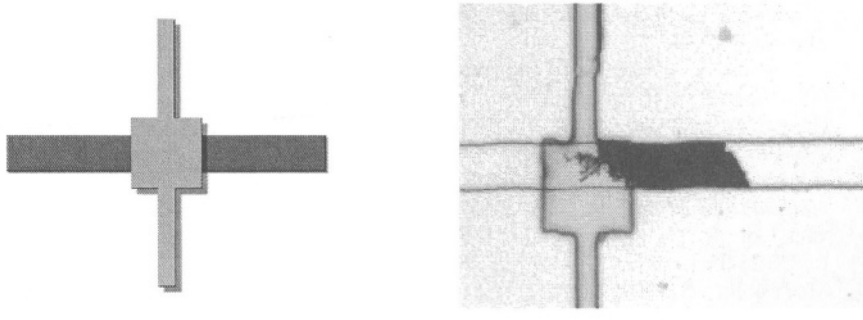


Fig. 6. (a) a schematic representation of a valve. The dark gray channel is the solution flow channel, while the light gray channel is the pneumatic valve over the flow channel separated by a 40 μm membrane, (b) 5.6 μm beads were collected before a partially closed valve. The flow channel is 100 μm and the valve pad is 200x200 μm

6 Conclusion

With new technologies emerging in the microfluidics field, more sophisticated systems can be designed and manufactured allowing reusable and reprogrammable DNA computers. This field will not only proof to be beneficial for DNA computing applications, but related applications as well, such as medical diagnostics. Negative selection is possible, with variable parameters such as the reactor and bead size (the size of the beads determines the surface increase) and starting concentration. By using a different architecture of the OR selector, it may be possible to skip the production of a prefabricated DNA-library and only manipulate the bit representations, allowing computations with more bits to be feasible.

Acknowledgement

The authors wish to acknowledge the support from DARPA award F30602-01-2-0560 to L. F. L. and NSF award 0121405 to Lydia L. Sohn and L. F. L. We would also like to thank Mike Livstone and Li Chin Wong for valuable suggestions.

References

[1] Braich, R. S., Chelyapov, N., Johnson, C., Rothemund, P. W. K., and Adleman, L. (2002) *Solution of a 20-Variable 3-SAT Problem on a DNA Computer.* Science **296**, 499-502.

[2] Faulhammer, D., Cukras, A. R., Lipton, R. J. & Landweber, L. F. (2000) *Molecular computation: RNA solutions to chess problems.* PNAS **97**, 1385-1389.

[3] Winfree, E. (2000) *Algorithmic Self-Assembly ofDNA: Theoretical Motivations and 2D Assembly Experiments.* Journal of Biomolecular Structure & Dynamics **11,** 263-270.

[4] Suyama, A. (2002) *Programmable DNA computer with application to mathematical and biological problems.* Preliminary Proceedings, Eigth International Meeting on DNA Based Computers, June 10-13, 2002, Japan, 91.

[5] Gehani. A. and Reif, J. (1999) *Micro flow bio-molecular computation* Biosystems **52**, 197-216.

[6] van Noort, D., Wagler, P. and McCaskill, J. S. (2002) *The role of microreactors in molecular computing.* Smart Mater. Struct. **11**, 756-760.

[7] van Noort, D., Gast, F.-U. and McCaskill, J. S. (2001) *DNA computing in microreactors.* LNCS 2340, 33-45.

[8] McCaskill, J. S. (2001) *Optically programming DNA computing in microflow reactors.* Biosystems **59**, 125-138.

[9] Marc A. Unger, Hou-Pu Chou, Todd Thorsen, Axel Scherer, Stephen R. Quake (2000) *Monolithic Microfabricated Valves and Pumps by Multilayer Soft Lithography,* Science **288**, 113-116.

In Vitro Translation-Based Computations

Yasubumi Sakakibara[1] and Takahiro Hohsaka[2]

[1] Department of Biosciences and Informatics, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522, Japan
yasu@bio.keio.ac.jp
[2] School of Materials Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa, 923-1292, Japan
hohsaka@jaist.ac.jp

Abstract. The translation system in a cell is a powerful molecular machine conducted by ribosomes, tRNAs, and several translation factors to synthesize proteins. The translation process is very accurate according to the genetic code and directed from 5'-end to 3'-end on messenger RNA. We employ the translation mechanism combined with four-base codon techniques to develop a molecular machine which computes finite automata (finite-state machine). We report some experimental results where we have succeeded to implement a finite automaton on an *E. coli in vitro* translation system with four-base codons.

1 Introduction

A fundamental operation used in most DNA computing methods is a "hybridization" which sticks two (single stranded) DNA molecules with Watson-Crick complementarity. For example, the self-assembly computation significantly makes use of hybridization operations. However, the single use of hybridizations is not stable and often leads to mis-hybridizations and imprecise computations. Many works have been devoted to design appropriate DNA sequences in order to avoid such mis-hybridizations.

In this paper, we present a completely different and novel approach to execute a precise DNA computation in a test tube. The translation system in a cell is a powerful molecular machine conducted by ribosomes, tRNAs, and several translation factors to synthesize proteins. The translation process is very accurate according to the genetic code and directed from 5'-end to 3'-end on messenger RNA. We employ the *in vitro* translation mechanism to execute precise hybridizations and precise DNA computations. Specifically, we implement finite-state automata using the *in vitro* translation system and four-base codon techniques.

Four-base codon methods [2] have been developed for position-specific incorporation of nonnatural amino acids into proteins through *in vitro* protein syntheses. The method extends the genetic code to allowing four-base codons such as AGGU and GGGU, and constructs chemically aminoacylated tRNAs containing complementary four-base anticodons by using T7 RNA polymerase. Such translation system with four-base codon could bring an effect of frame shifting.

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 197–202, 2004.

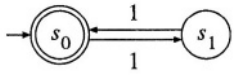


Fig. 1. A simple finite automaton of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepting input strings with even numbers of 1 symbols and rejecting input strings with odd numbers of 1s.

Our main idea to implement finite-state automata using *in vitro* translation system and four-base codons is that an input string is encoded into an mRNA in a specific manner, state-transition rules for a given finite automaton are represented by three-base codons, four-base codons and their combinations, and a computation (accepting) process of the automaton for an input string is simulated by the translation system.

In the next section, we describe the method in details using an example of simple finite automaton, illustrated in Figure 1, which is of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepts input strings with even numbers of 1 symbols and rejects input strings with odd numbers of 1s.

2 Methods

2.1 Implementing Finite-State Automata

The input symbol '1' is encoded to the four-base subsequence GGGU and an input string is encoded into an mRNA by concatenating GGGU and A alternately and adding AAUAGC at the 3'-end. This one-nucleotide A in between GGGU is used to encode two states $\{s_0, s_1\}$, which is a same technique presented in [4]. For example, a string "111" is encoded into an mRNA:

$$\underbrace{\text{GGGU}}_{1}\,\text{A}\,\underbrace{\text{GGGU}}_{1}\,\text{A}\,\underbrace{\text{GGGU}}_{1}\,\text{AAAUAGC}.$$

The four-base anticodon (3')CCCA(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')UCC(5') and (3')CAU(5') encodes the rule $s_1 \xrightarrow{1} s_0$. Further, the encoding mRNA is linked to GFP-coding RNA subsequence as a reporter gene for the detection of successful computations. Together with these encodings and tRNAs containing four-base anticodon (3')CCCA(5'), if a given mRNA encodes an input string with odd numbers of 1 symbols, an execution of the *in vitro* translation 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 fluorescent signal of GFP. Examples of accepting processes are shown in Figure 2: (Upper) For an mRNA encoding a string "1111", the translation successfully goes through the entire mRNA and

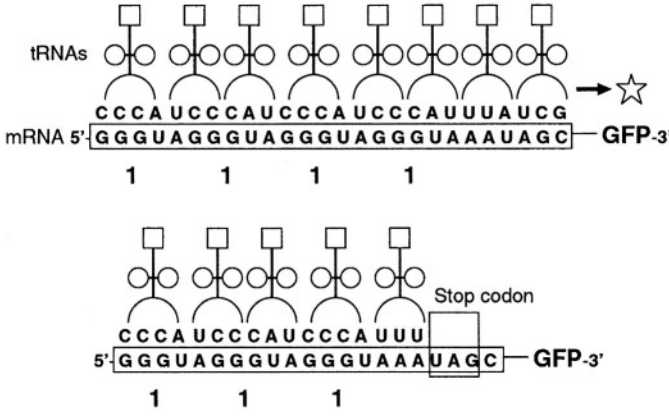


Fig. 2. Examples of accepting processes: (Upper) For an mRNA encoding a string "1111", the translation successfully goes through the mRNA and translates the reporter gene of GFP emitting the fluorescent signal. (Lower) For an mRNA encoding a string "111", the translation stops at the stop codon UAG, does not reach to the GFP region and produces no fluorescent signal

translates the reporter gene of GFP which emits the fluorescent signal. (Lower) For an mRNA encoding a string "111", the translation stops at the stop codon UAG, does not reach to the GFP region and produces no fluorescent signal.

If the competitive three-base anticodon (3')CCC(5') comes faster than the four-base anticodon (3')CCCA(5'), the incorrect translation (computation) immediately stops at the following stop codon UAG.

2.2 Four-Base Codons and in Vitro Translations

Four-base codon methods [2] extends the genetic code to allowing four-base codons such as AGGU and GGGU, and constructs chemically aminoacylated tRNAs containing complementary four-base anticodons by using T7 RNA polymerase. The four-base codons were successfully decoded by the the nitrophenylalanine-tRNA containing the complementary four-base anticodons in an *E. coli in vitro* translation system.

The plasmids pGFPG3TA$_n$ containing $(\mathtt{GGGTA})_n$ sequence between T7-tag sequence and EGFP gene are prepared according to a standard protocol for site-directed mutagenesis. The coding regions of the pGFPG3TA$_n$ are amplified by PCR, and then used as a template for T7 RNA polymerase reaction. The mRNAs obtained by the T7 RNA polymerase reaction are purified by ethanol precipitation. The aminoacyl-tRNA with (3')ACCC(5') anticodon is prepared as described in [2]. Briefly, the tRNA lacking 3' CA dinucleotide is prepared by the T7 RNA polymerase reaction using synthetic tRNA gene as a template. On the other hand, the CA dinucleotide that is aminoacylated with *p*-nitrophenylalanine is chemically synthesized. Then, the tRNA(-CA) and the

nitrophenylalanine-dinucleotide are linked together by T4 RNA ligase. The resulting nitrophenylalanine-tRNA is isolated by ethanol precipitation.

The *in vitro* translation is carried out as previously described in [2]. The mRNA encoding T7-tag, $(\mathtt{GGGUA})_n$, and EGFP is added to an *E. coli in vitro* translation system in the presence of the nitrophenylalanine-tRNA with the (3')ACCC(5') anticodon. The reaction mixture is incubated at 37°*C* for 1 hour, and applied to denatured SDS-PAGE and Western blotting using anti-T7tag antibody.

2.3 General Theory to Implement Finite Automata Using n-Base Codons

A general theory behind the *in vitro* implementation of finite automata presented in Section 2.1 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 vitro* translation system 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 [4]. 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 vitro* translation system to stop a translation if the finite automaton does not accept an input string.

3 Experiments

We have done some laboratory experiments for the finite automaton shown in Figure 1, which is of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepts

Table 1. Three sequences of $GFPG3TA_n$ $(n = 1, 2, 3)$ for experiments

Input string	Coding sequence of $GFPG3TA_n$
"1"	[T7tag] - GAATTC(*Eco*RI) - GGGTA - AATAGC - [EGFP]
"11"	[T7tag] - GAATTC(*Eco*RI) - GGGTAGGGTA - AATAGC - [EGFP]
"111"	[T7tag] - GAATTC(*Eco*RI) - GGGTAGGGTAGGGTA - AATAGC - [EGFP]

input strings with even numbers of 1 symbols and rejects input strings with odd numbers of 1s, by using the four-base codon and the translation system shown in Section 2.2.

We tested our method for three input strings, "1", "11", and "111", to see whether the method correctly accepts the input string "11" and rejects the strings "1" and "111". For these experiments, we used three well-designed sequences of $GFPG3TA_n$ shown in Table 1.

The products of the *in vitro* translation reaction were analyzed on *Western blotting* using an antibody against N-terminal T7tag sequence. The results are shown in Figure 3. In the case of $GFPG3TA_1$, no full-length protein was produced. On the other hand, $GFPG3TA_2$ gave the full-length protein observed at the same position as the wild-type EGFP. When $GFPG3TA_3$ was used as a template, no full-length band was observed as the case of $GFPG3TA_1$. In any cases, no full-length protein was produced in the absence of nitrophenylalanine-tRNA with (3')ACCC(5') anticodon. These results indicate that the full-length GFP was synthesized when a duplicate GGGTA sequence was introduced, but was not synthesized when a single or triplicate GGGTA was introduced. Therefore, the *in vitro* translation system correctly computed the finite automaton to accept one input string "11" with even numbers of 1s and to reject two input strings "1" and "111" with odd numbers of 1s. Also, the aminoacyl-tRNA with (3')ACCC(5') anticodon was essential for the correct translation.

4 Discussions

A related work which aims to implement finite automata on DNA computers is Shapiro et al. [1] that have successfully implemented the finite-state machine by the sophisticated use of the restriction enzyme (actually, *Fok*I) which cut outside of its recognition site in a double-stranded DNA. Our method adopts a completely different strategy to implement finite automata in text tube from their method and very original.

Since the finite automata are an useful technique often used for sequence analyses in silico such as BLAST, one potential application of our method is *in vitro* sequence analyses. Our *in vitro* automata could directly examine expressed mRNAs in test tube instead of sequencing and converting them onto electric data. We will report this research direction in more details.

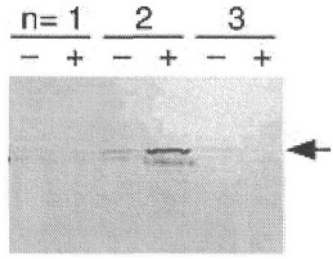


Fig. 3. Western blot analysis of the *in vitro* translation products using GFPG3TA$_n$ ($n = 1, 2,$ and 3) as templates in the absence (–) or presence (+) of nitrophenylalanine-tRNA with (3')ACCC(5') anticodon. A portion of the *in vitro* translation reaction mixture was applied to SDS-polyacrylamide gel electrophoresis followed by Western blotting. Translation products containing T7tag sequence were visualized on the blot. An arrow indicates the full-length EGFP protein

Acknowledgements

This work is supported in part by Grant-in-Aid for Scientific Research (C) No. 13680464 and Grant-in-Aid for Scientific Research on Priority Area No. 14085205. 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.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

Autonomous Biomolecular Computer Modeled after Retroviral Replication

Nao Nitta [1] and Akira Suyama [1,2]

[1] Department of Life Sciences,
Graduate School of Arts and Sciences, The University of Tokyo,
3-8-1 Komaba, Meguro-ku, Tokyo 153-8902, Japan
nitta@genta.c.u-tokyo.ac.jp
[2] Institute of Physics,
Graduate School of Arts and Sciences, The University of Tokyo,
3-8-1 Komaba, Meguro-ku, Tokyo 153-8902, Japan
suyama@dna.c.u-tokyo.ac.jp

Abstract. We designed a retroviral computer, of which hardware is composed of DNA/RNA dependent DNA polymerase, transcriptase, RNaseH, and DNA and RNA strands. Sequences of DNA strands define functions and RNA single strands work as arguments and return values for each function. In this paper, we show that computational jobs, such as encoding of input data and AND/OR operation, can work on this computer. By combining multiple functions, more complex molecular programs for gene analysis can be constructed. Experimental study showed that some functions were actually executed *in vitro* autonomously. Since this computer has originally derived from the retrovirus mechanism, we expect an *in vivo* computer will be realized from this technology, which detects the cell state through gene expression patterns, and controls the cell conditions with output RNA. It may provide a powerful tool for both research and clinical application.

1 Introduction

At the beginning of DNA computer history, it attracted attention because of its potential to realize parallel computer that can be applied to solve NP-complete problems [1, 2]. Application of DNA computer to a larger scale computation was recently reported [3, 4], however, it is still difficult to overcome the conventional electric computer. A recently occurred, interesting research field around the molecular computing is its application to biological purposes, such as gene analysis and diagnosis [5-9]. This seems reasonable because it can use biomaterials directly as input data, and compute in molecular level.

Here we report a molecular computer that suits for biological purposes. By modeling the retroviral replication mechanism, we created a programmable computer composed of biomolecules, which works autonomously under an isothermal condition. In section 2, we shall describe the advantage of referencing retrovirus genomic replication, and describe the architecture and mechanism of the molecular computer in section 3. Hardware of this computer consists of adequate enzymes, DNA and RNA. Sequences of DNA strands define the functions used in the

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 203-212, 2004.

computation processes, and RNA molecules act as arguments and return values. By mixing multiple functions, complex program can be created. We shall present a method for making gene analysis programs with logic operations in section 4. Experimental results in section 5 demonstrate that some functions actually work autonomously under an isothermal condition. This technology has a high potential to realize a biomolecular computer, which works not only *in vitro* but also *in vivo.*

2 Autonomous Molecular Computer and Retrovirus

DNA computer uses DNA molecules either as data containers or as computational programs. Almost all DNA computers require physical binding of DNA strands, i.e., binding of primers or probes to their target sequences, to access data coded on the DNA sequences. When programs are executed, in most DNA computer systems, repeated access to data set is needed. Once data access has done, stable double-stranded DNA has been formed. So that some following processes, which create open-to-access single-stranded DNA from closed double-stranded DNA, is essential. In many previous works, this data-releasing process was pursued by unwinding double-stranded DNA through heating, which requires external control.

To create a molecular computer that works autonomously under an isothermal condition, molecular reactions that release 'open' sequences from 'closed' double-strand sequences are essential. A recently reported method to realize an autonomous DNA computer uses restriction enzymes to create the open-to-access single-stranded target DNA from the closed double-stranded DNA [10]. Restriction enzymes, however, create single-stranded sticky ends of only several bases. Available data set variety is thus limited. As for a molecular computer for biological application, it should flexibly accept any sequences as input data because assumed input variety from biological sample is vast.

Here we focused on the retrovirus genome replication system. Retrovirus genome is composed of single-stranded RNA molecule. Inside a host cell, the RNA genome is reverse transcribed into single-stranded DNA. Concurrently, the template RNA strand is destroyed with RNaseH activity. Then double-stranded DNA is synthesized, and many copies of the viral genomic RNA are made by transcription. From an informational viewpoint, it is an RNA input/output device returning the same output sequence as the input. In this process, RNaseH reaction and transcription act as the data-releasing steps that produce the open-to-access single-stranded nucleic acid. By imitating the retroviral replication system, an autonomous molecular computer that suits for biological application is likely to be realized.

3 Computer Architecture

3.1 Hardware and Molecular Reactions

The computer hardware is composed of the reaction solution with DNA, RNA and four enzymatic activities, RNA dependent DNA polymerase (reverse transcriptase), RNaseH, DNA dependent DNA polymerase, and DNA dependent RNA polymerase

(transcriptase). Programs that work in this hardware are described in DNA sequences as combination of multiple functions. This hardware is equivalent to the reaction solution of the self-sustained sequence replication (3SR; [11]) or the cooperatively coupled *in vitro* amplification system (CATCH; [12]), both of which are the RNA based gene amplification method.

3.2 Functions

Function, in general, is a relation between arguments and return-values, which return-values are dependent on, and uniquely decided from, arguments. Functions of the present molecular computer take RNA molecules as arguments, execute set of molecular reactions on the computer hardware, and output RNA as return-values. Combination and relation between arguments and return-values are determined by sequences of DNA strands defining functions.

Figure 1 shows a schematic view of the function. The target specific region at the 3' end of primer binds to argument RNA and creates first strand complementary DNA (cDNA) by reverse transcription. Simultaneously RNaseH destroys the template RNA, then double stranded DNA is created. When promoter sequence is located on the primer DNA, transcription occurs from the double stranded promoter. Only double stranded promoter sequences can initiate transcription. Synthesized RNA acts as the return-values for the function.

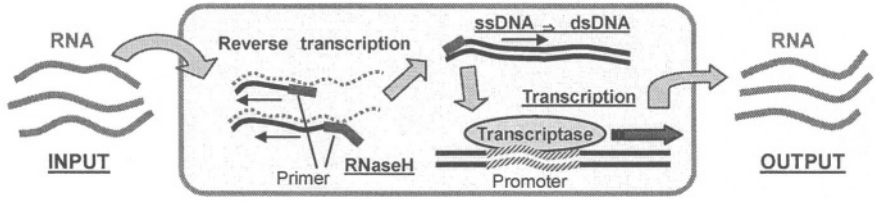


Fig. 1. Schematic view of 'function'

The most fundamental functions consisted of two primer DNA fragments, at least one of which contains the promoter sequence. One primer initiates reverse transcription, and the other works to synthesize double stranded DNA. Arguments for the function are determined by their 3' end sequences. The location of the promoter sequence determines the type of function, and the sequence downstream decides a return-value. Figure 2A shows one example of such functions. Here, 'Ta' on the first primer and 'Tb' on the second primer are target specific sequences, and the promoter sequence is located on the first primer. 'P' and 'Q' are optional sequences. Bars on the top of letters indicate reverse complement sequence. When adequate input RNA is given to this function, the first and second strand cDNAs are synthesized, and then RNA molecule consisting of the reverse sequence of input RNA and sequences P and Q attached at the 5' and 3' end is returned.

Note that the target sequence can be separated on multiple RNA molecules. When the 5' end of the first RNA is equal to the sequence somewhere on the second RNA, reverse transcription succeed from one to next, and track sequences on multiple RNA molecules (reverse-transcriptional pathway). In such a case, this function returns the

reverse sequence of reverse-transcriptional pathway from Ta to Tb, attached by P and Q.

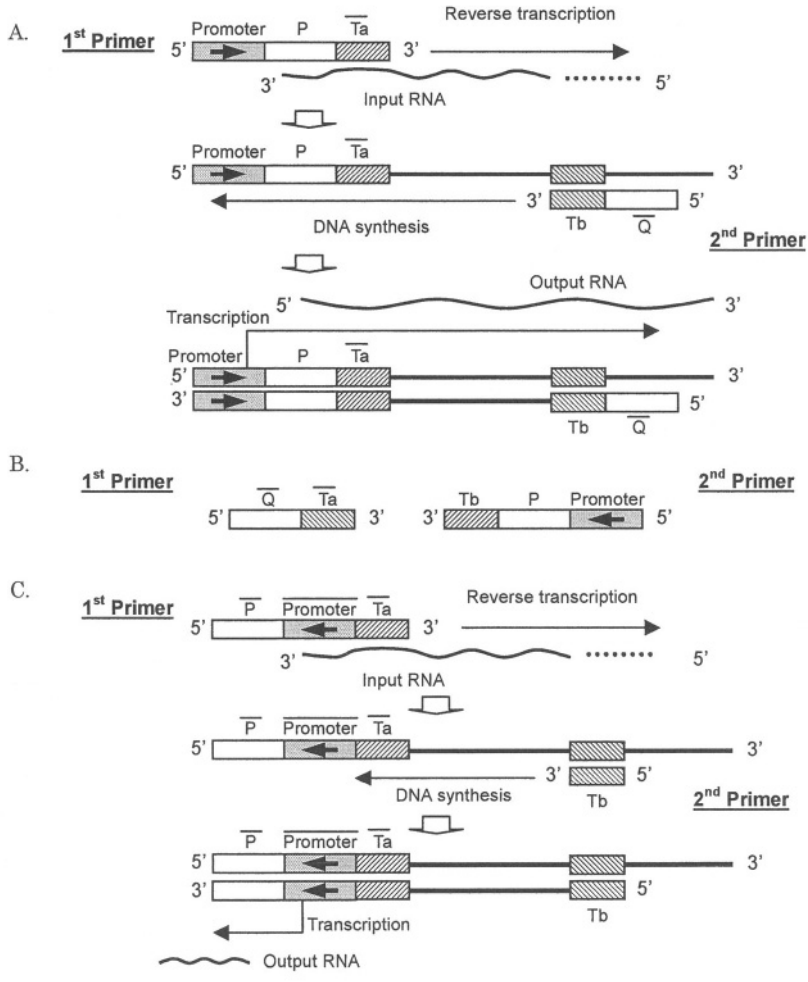


Fig. 2. Examples of functions composed of two DNA primers

Other kinds of functions can be generated by altering location and direction of the promoter. Figure 2B shows the case that promoter sequence is located on the second primer. This function works when the reverse-transcriptional pathway that initiates at Ta and ends Tb exists, and returns sequence of the pathway. Figure 2C is another example that direction of promoter sequence inverted, and this function returns sequence P when reverse-transcriptional pathway from Ta to Tb exists.

3.3 Programs

Since arguments and return-values of each function consist of RNA molecules, it is presumably possible to combine multiple functions, and build more complicate programs. When plural functions work in a single solution at once, since all RNA molecules can be arguments for all functions in the solution, it is a kind of amorphous computer. We present one example of the program for gene expression analysis in the next section.

4 Programs for Gene Analysis

4.1 Gene Encoding

Since there are a huge variety of gene sequences in organisms, the gene-encoding process that converts the target gene sequence into the corresponding internal code sequence is absolutely essential to develop a molecular computer program generally available for gene expression analysis [5, 6]. On the present molecular computer, the encoding can be done with a function indicated in Fig. 2C. Ta and Tb are specific sequences to the target gene, and P is its corresponding internal code. Any sequence is available as internal code, but preferably sequences suitable for subsequent logic operations. Some desirable characters are, combine well only with the target and do not stably bind with unexpected sequences, do not form stable hairpin structures, and thermal stability and length are uniform [13].

4.2 AND/OR Operation

Figure 3 shows the structure of a program of the AND operation. First, the target gene is encoded into the corresponding internal code RNA of 5'-[H]-[C2]-[H]-[Cl]-3' (Fig. 3 A). C1 and C2 are the internal code sequences, and H is a part of promoter sequence that always attached on the head of the transcribed RNA.

To execute AND operation between two genes, two types of functions are used. The first type of functions can encode one target gene (gene A) to 5'-[H]-[C2]-[H]-[Cl]-3' and the other gene (gene B) to 5'-[H]-[C3]-[H]-[C2]-3'. The second type of function returns sequence X when reverse-transcriptional pathway from C1 to C3 exists (Fig. 3B). Combining these functions all together, a program, which performs the AND operation between two target gene and returns RNA of sequence X, is made. Figure 3C shows the framework. Blank arrows represent functions, numbered boxes represent codes and black arrows indicates reverse-transcriptional pathways.

OR operation is easier, and one example is to encode two genes into a same code, or a corresponding pathway.

4.3 Gene Expression Pattern Analysis with Logic Operation

A point of the logic operation of genes is to encode each gene into a fragment of reverse-transcriptional pathway. By forming larger pathway, it is theoretically possible to execute more complex logic operation. For example, in Fig. 3D, Gene A is

encoded to pathway 1->2, Gene B to 2->3... and so on, and put a function that returns Gene X when pathway from 1 to 6 exists. This program, as a whole returns Gene X when ['Gene A' AND {('Gene B' AND 'Gene C') OR ('Gene D' AND 'Gene E' AND 'Gene F')}] is true.

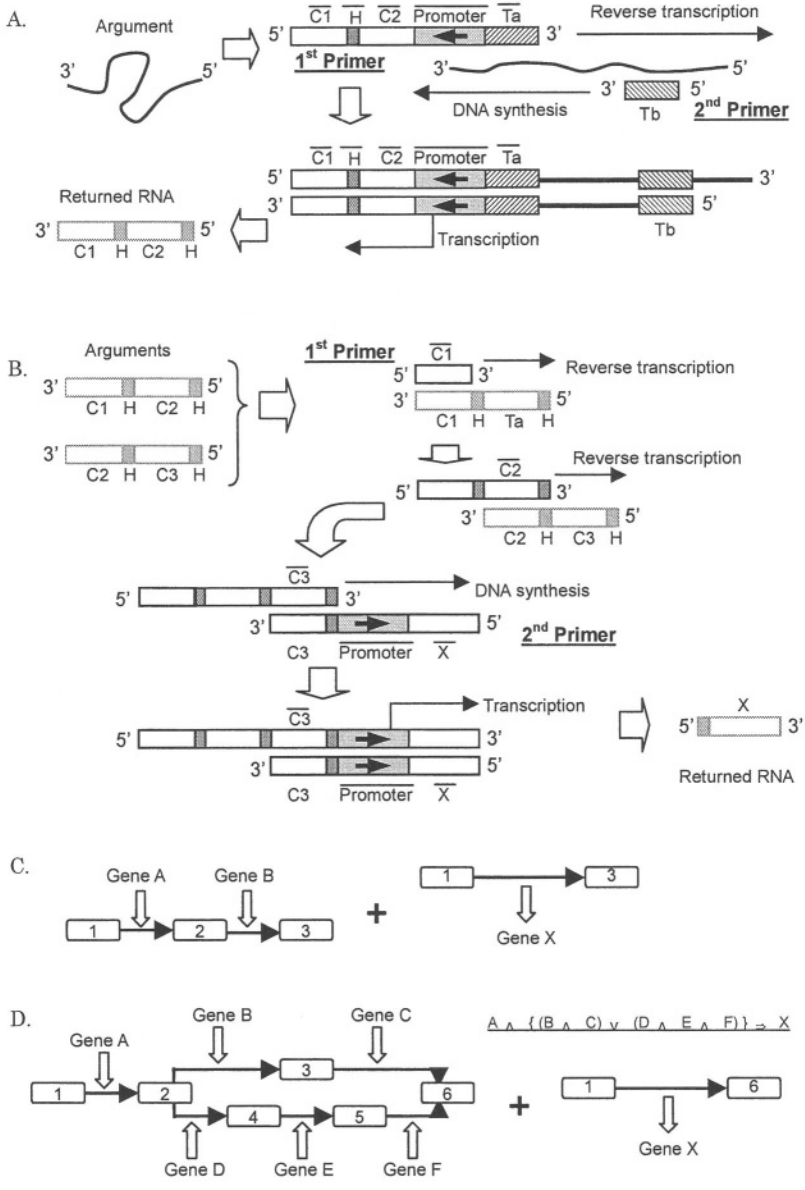


Fig. 3. A) Encode gene to internal code that suits for logic operation. B) Reactions in AND operation. C) Framework of the AND operation. D) Example of complex logic operation program for gene analysis

5 Experimental Results

5.1 Materials and Methods

DNA primers are synthesized by Qiagen, and molecules used as input are synthesized with *in vitro* transcription. DNA sequences used here are listed below.

[TGTP-PT] 5'- GAT GCA TAA TAC GAC TCA CTA TAG GGA GAG GGG ATG AAT TTC TAC TTT G-3'; [TGTP-AR] 5'- GC TTG TCT TCT AAG GAC TCA TCA TTG -3'; [TGTP-P1] 5'- CTG AGG TTA TCT TGG TCT GGG GAG A T CTC CCT ATA GTG AGT CGT ATT ACT GAG GTT ATC TTG GTC TGG GGA GAC AGA TAT ATA TGG TCC CAC C -3'; [aT21] 5'- ATA GGG AGA GAC AAA CAC CCC GAA TAC AAA CAG CGG GAG ATG AAG TCA CCA CAA CAC ACA GTA CA -3'; [TGTP-S2] 5'- ACT TAC TAT CGC ATG GCT TA -3'

RNA input and DNA primers were mixed in the reaction buffer composed of 40 mM Tris-HCl (pH 8.0), 50 mM NaCl, 8 mM $MgCl_2$ and 5 mM DTT. After a 5 min incubation at 65°C, 0.3 U/μl AMV Reverse Transcriptase XL (Takara Bio, Otsu, Japan), 0.04 U/μl Ex Taq™ (Takara Bio) and 3.2 U/μl Thermo T7 RNA Polymerase (Toyobo, Osaka, Japan) were added to the reaction mixture. It was then incubated at 50°C for a computational reaction.

To detect the RNA in the reaction solution as a result of the computation, the reaction solution was heated at 85°C to 10 min to inactivate transcriptase and filtered with Microcon YM-100 (Millipore) to remove enzymes. It was then concentrated with Microcon YM-10 (Millipore) and treated with Deoxyribonuclease I (Amplification Grade; Invitrogen) to remove primers and intermediate DNA and with AMV Reverse Transcriptase to XL (Takara Bio) to synthesize complementary DNA. The cDNA was quantitatively analyzed by a real-time PCR method using LightCycler™ (Roche Diagnostics). The PCR products of the cDNA were analyzed by electrophorese on Agilent 2100 bioanalyzer (Agilent Technologies). All these handlings followed the manufacturers' protocol.

5.2 Gene Amplification Function

To demonstrate the computer hardware works well, we performed a gene amplification function that resembled 3SR [11]. It is a type of function illustrated in Fig. 2B, with Ta and Tb as target gene specific sequences. Here, the *in vitro* expressed TGTP RNA was selected as a target gene. TGTP-AR was used as the first primer that initiates reverse-transcription of the TGTP RNA, and TGTP-PT was used as the second primer. The TGTP-PT contains T7 promoter sequence at its 5' end, and the target RNA specific sequence at the 3' end. This function returns TGTP fragment when the target TGTP RNA exists. Since the output of the function recursively call the function, as a whole, it works as a target gene amplification function.

Experimental results showed that this function worked specifically (Fig. 4). Amplification was observed within 15 minutes after the input of TGTP gene, but no signal was observed with the Vitronectin gene input or no input RNA.

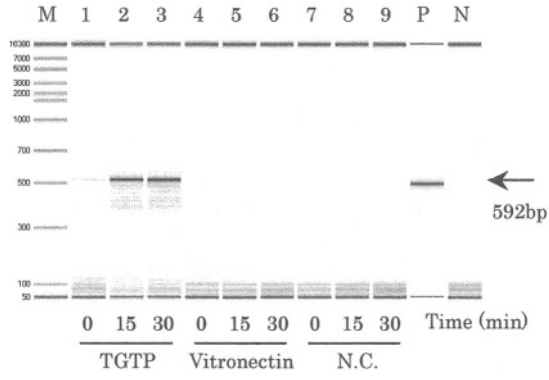


Fig. 4. Result of TGTP gene amplification function (M: Marker, P; Positive, N; Negative). When incubated with TGTP gene, amplification of 592bp RNA was observed (lane 1-3), but no signal with Vitronectin (4-6) or without input (7-9)

5.3 Encoding Function

Encoding processes from genes to internal codes are the most important parts in gene analysis programs. We experimented the encoding of a TGTP gene into its corresponding internal code. The structure of primers are indicated in Fig. 3A, with Ta and Tb as TGTP gene specific sequences. TGTP-P1 and TGTP-S2 were used as primers. Oligo DNA aT21, which binds to the code sequence on TGTP-P1, was also added to the reaction mixture to inhibit binding of returned code RNA to yet unreacted primers. This inhibition is important to increase the efficiency of the encoding reaction because RNA strands of DNA-RNA hybrids are destroyed by RNaseH activity. A result in Fig. 5 indicates that the encoding was successfully executed. The output increased to the maximum level within 30-40 min after the input of target gene TGTP.

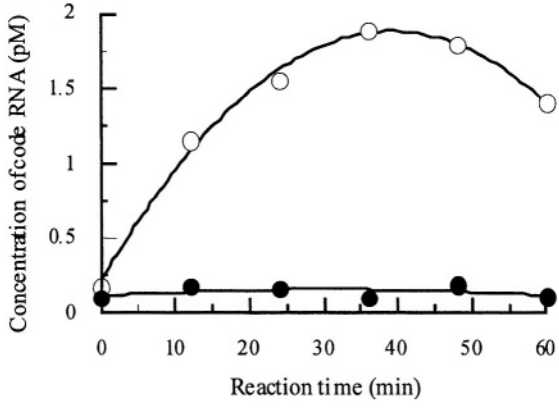


Fig. 5. Encoding of TGTP gene into internal code. TGTP gene was incubated with the encoding function (○). Code RNA was produced and reached the peak in 30-40 minutes incubation. Code RNA was not increased when TGTP was not added (●)

6 Discussion

Viruses inject minimized components into host cells, and execute a set of reactions to replicate themselves. Even though the detailed system varies, it can be viewed that all kinds of viruses utilize host cells as computer hardware, and run a self- proliferating program. Our main motivation is to create a molecular computer with a potential of working inside living cells. The retrovirus replication mechanism will give us a good clue to design such a molecular computer.

Theoretical study in this paper showed it is possible to construct an autonomous molecular computer with enzyme activities used in retroviral genomic RNA replication, and that the computer can be really practical for biological purpose although it is unlikely to be suitable for mathematical use. Here we presented a gene analysis program with the capability of logical operation. The present architecture can also apply to other probable program structures including a neural network.

In the experiment, we used TGTP gene, which is available as a marker gene for graft-versus-host disease (GVHD) [14]. Our experimental result showing the capability of gene expression analysis demonstrates that the present molecular computer can provide a powerful tool for medical diagnosis. We also tried to perform multi-function programs, but they are not yet succeeded. Modifications of hardware to increase the efficiency of computation reactions are required to make this computer more reliable and practical.

Our ultimate goal is to create an *in vivo* molecular computer. The computer presented here potentially works inside living cells. There may, however, be some problems to be solved to realize it, such as how to introduce all components required for computation into living cells, and how to work them properly intra-cellular environment. By overcoming all these problems, we expect that a new technology will occur, which detects the cell state through gene expression patterns, and controls the cell with output RNA. It may provide a powerful tool for both research and medical purposes.

Finally, it is also interesting to inquire the hypothesis that such a computational process naturally exists, since there are huge number of RNA molecules observed inside mammalian cells, whose functions are not yet clarified.

References

[1] Adleman, L.M.: Molecular computation of solutions to combinatorial problems. *Science* **266**(1994)1021-1024

[2] Lipton, R.J.: DNA solution of hard computational problems. *Science* **268** (1995) 542-545

[3] Braich, R.S., Chelyapov, N., Johnson, C., Rothemund, P.W., Adleman, L.: Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* **296** (2002) 499-502

[4] Nakajima, T., Sakai, Y., Suyama, A.: Solving a 10-variable 43-clause instance of 3-SAT problems on DNA computer automatically executing a basic instruction set. *Preliminary Proc. of The Eighth International Meeting on DNA Based Computers* (2002) 332

[5] Suyama, A., Nishida, N., Kurata, K., Omagari, K.: Gene expression analysis by DNA computing. *Currents in Computational Molecular Biology 2000 (S. Miyano, R. Shamir, T. Takagi, Eds), Universal Academy Press, Inc., Tokyo, Japan* (2000) 12-13

[6] Nishida, N., Wakui, M., Tokunaga, K., Suyama, A.: Highly specific and quantitative gene expression profiling based on DNA computing. *Genome Informatics* **12** (2001) 259-260
[7] Mills, A.P. Jr.: Gene expression profiling diagnosis through DNA molecular computation. *Trends Biotechnol.* **20** (2002) 137-140
[8] Morimoto, N., Kiyohara, H., Sugimura, N., Karaki, S., Nakajima, T., Makino, T., Nishida, N., Suyama, A.: Automated processing system for gene expression profiling based on DNA computing technologies. *Preliminary Proc. of The Eighth International*
[9] *Meeting on DNA Based Computers* (2002) 331
Normile, D.: DNA-based computer takes aim at genes. *Science* **295** (2002) 951
[10] Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules. *Nature* **414** (2001) 430-434
[11] Guatelli, J.C., Whitfield, K.M., Kwoh, D.Y., Barringer, K.J., Richman, D.D., Gingeras, T.R.: Isothermal, *in vitro* amplification of nucleic acids by a multienzyme reaction modeled after retroviral replication. *Proc. Natl. Acad. Sci. USA.* **87** (1990) 1874-1878
[12] Ehricht, R., Kirner, T., Ellinger, T., Foerster, P., McCaskill, J.S.: Monitoring the amplification of CATCH, a 3SR based cooperatively coupled isothermal amplification system, by fluorimetric methods.*Nucleic Acids Res.* **25** (1997) 4697-4699
[13] Yoshida, H., Suyama, A.: Solution to 3-SAT by breadth first search..In DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society **54** (2000) 9-22
[14] Wakui, M., Yamaguchi, A., Sakurai, D., Ogasawara, K., Yokochi, T., Tsuchiya, N., Ikeda, Y., Tokunaga, K.: Genes highly expressed in the early phase of murine graft-versus-host reaction. *Biochem. Biophys. Res. Commun.* **282** (2001) 200-206

Biomolecular Computing by Encoding of Regulated Phosphorylation-Dephosphorylation and Logic of Kinase-Phosphatase in Cells

Jian-Qin Liu and Katsunori Shimohara

ATR Human Information Science Laboratories,
2-2-2 Hikaridai, "Keihanna Science City", Kyoto, 619-0288, Japan
{jqliu,katsu}@atr.co.jp

Abstract. As the first step in studying cell-based computing, a new method of biomolecular computing by cells is proposed based on signaling pathways of kinases and phosphatases for phosphorylation-dephosphorylation (we call this method "kinase computing" for short in the latter parts of this paper). As opposed to the Adleman-Lipton paradigm of DNA computing and other types of cell-based computing, the core mechanism of kinase computing that carries out recursive computation at the biological level is based on (1) encoding the information by phosphorylation and dephosphorylation, (2) running the selection operators by coupled pathways of kinases and phosphatases under certain conditions, and (3) readout by immunofluorescence analysis. The control schemes for the related synchronization processes in 3-SAT computation is studied to clarify the biological feasibility of kinase computing, in which the control-space complexity and time complexity are linear.

1 Introduction

The materials for building biomolecular computers vary among DNA, RNA, proteins, enzymes, cells and other biomolecules. Why we select cells as the material for building biomolecular computers is the question we have to answer. Successful examples of cell-based computation include ciliates-based cell computing by L. Landweber and L. Kari [1] and computing in ciliates by A. Ehrenfeucht et al. [2], amorphous computing by R. Weiss and T. Knight [3], membrane computing (P-systems) by G. Păun [4], and others (e.g., [5]). Our answer is that among different materials for building molecular computers, cells are excellent objects that possess a high internal spatial complexity to compensate for the complexity caused by the problem to be solved, where the related signaling transduction mechanism can provide the ability to control the related biochemical processes by only using a limited number of enzymes. From the fact that cells function in the above-mentioned way to shelter the signaling mechanism with related pathway regulation, we propose a new method of biomolecular computing based on the signaling pathways of cells. In this method, we only need to control a limited number of kinases and phosphatases, which is much smaller than the exponential number of manufactured DNA molecules. By regulating an efficient number of molecules (kinases and phosphatases),

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 213-218, 2004.

computation is arranged in terms of pathways in cells so that it can be applied to NP problem solving. This method is expected to be implemented by the signaling pathways of phosphorylation and dephosphorylation guided by Rho family GTPases of mammalian cells. This differs from the Adleman-Lipton paradigm of DNA computing, surfaced-based techniques, and other cell-based computing methods.

2 3-SAT Computing by Phosphorylation-Dephosphorylation

With the goal of achieving consistency between simulation results and the molecular mechanism of signaling known from evidences the operations of signaling pathways guided by Rho family GTPases were constructed for 3-SAT computation. The phosphorylation-dephosphorylation encoding by molecular mixtures and the kinase-phosphatase selection by pathways are the two major factors. In this section we discuss how these signaling processes of cells are used for 3-SAT computation in concrete molecular objects within the cell communications under the regulation of Rho family GTPases. From the simulation, we can observe quantitative measures such as "concentration vs. time" curves while comparing them with the corresponding biochemical reaction in related functions. Stable states and synchronization should be guaranteed so that the 3-SAT computation can be successfully carried out. From the studies of the empirical parameters' setting and related theoretical analysis, the parameters can be set to fit the requirements of the "regulated" signaling mechanism of cells. As for the possible biological implementation schemes, the biological operations can be made based on phosphorylation-dephosphorylation representation and kinase-phosphatase' pathway selection. In terms of the operations at a high (conceptual) level, we have developed a protocol for above-mentioned kinase computing which is composed of the operation set of Q_k = {make, selection, labeling, readout}. Based on engineered phosphorylation and dephosphorylation processes as well as comparisons with the biological operations at the micro level, the functions of the operations (operators) at a high (conceptual or macro) level can be briefly defined as follows: (1) make: to activate the population (i.e., the set of candidates) in the concerned cells; (2) selection: to select these candidates in the pathways; (3) labeling: to label the valid candidates through the pathways encoded for the constraints of the computation; (4) readout: to detect the solution from different pathways according to the threshold for common outputs and to get the final result. The integration of these operations is accomplished by the matter flow in cells with controlling schemes under the nutrient conditions that reflect the idea of the MIMD architecture. This is the core of the entire selection process constructed by the pathways of cells. The readout, which can be made by immunofluorescence analysis, mainly depends on the chemical sensors with certain accuracy. Here the key to selection is the kinase-phosphatase's switching in the manner of logic units.

2.1 Representation and Operations

At an abstract (conceptual) level, 3-SAT computation is constructed by the mechanism of signal transduction based on kinases and phosphatases. This computing

process is demonstrated through application to an instance of 3-SAT problem solving. A 3-SAT problem can be described as: let $\Psi_i = C_{i1} \vee C_{i2} \vee C_{i3}$ be a clause (i = 0, 1, ..., m) so that all clauses $\{\Psi_i\}$ produce the constraint Ξ with the form $\Psi_1 \wedge \Psi_2 \wedge \ldots \wedge \Psi_m$. Our task is to find the set of combinatorial forms of n variables that satisfies Ξ. We define that Mo[p-X_{ij}] (i.e., Mo[X_{ij}]) refers to the molecule that is encoded as the positive form of X_{ij} by attaching a phosphate to Mo[X_{ij}]; PATH (p-X_{ij}) refers to the signaling pathway that can accept the molecule Mo[p-X_{ij}]; Mo[dp-X_{ij}] (i.e., Mo[$\neg X_{ij}$]) refers to the molecule that is encoded for the negative form of X_{ij} (i.e., $\neg X_{ij}$) with the state of dephosphorylation (without any phosphate); PATH (dp-X_{ij}) refers to the signaling pathway that can accept the molecule Mo[dp-X_{ij}]. The input of the signaling pathways is $\{\Psi_i\}$. The output produces the molecules that are encoded for the solutions.

2.2 An Instance of Computing by the Rho-MBS-MLC Pathway

As an example for studying the entire process of kinase computing guided by the Rho family GTPases and corresponding signaling pathways, we constructed a sub-unit of 3-SAT computing by the regulated "Rho-MBS-MLC" pathway. Here, the "Rho-MBS-MLC" pathway refers to the pathway of phosphorylation-dephosphorylation, which consists of three main sub-pathways of Rho kinase, MBS, MLC and other related biological chemical reactions in cells. Here, we take a (2,1) sub-unit of the 3-SAT computing process as an instance to discuss the biologically faithful schemes in which the variables are x_1 and $\neg x_2$ and the clause is $x_1 \vee \neg x_2$, in which the bio-chemical reactions for 3-SAT computation are constructed by phosphorylation-dephosphorylation pathways. Through activating the functional proteins in membranes of cells by target molecules in inter-cell communications, the related engineered pathways regulated by the Rho-MBS-MLC pathways are employed to produce candidates with the phosphorylation-dephosphorylation representation. Here, we can get the set of {MBS--MLC, MBS--(MLC-p), (p-MBS)--MLC, (p-MBS)--(MLC-p)} corresponding to the set of {00, 01, 10, 11}, which is implemented by binding the two types of molecules, where (p-MBS) for "x_1" refers to "1" in digital and "T" in logic; MBS for "$\neg x1$" refers to "0" in digital and "F" in logic; MLC-p for "x_2" refers to "1" in digital and "T" in logic; MLC for "$\neg x2$" refers to "0" in digital and "F" in logic; "label protein" is denoted as L. For clause 1 of $(x_1 \vee \neg x_2)$, we can design the following two pathways for selection: Pathway 1 is used to realize the first part "x_1" and pathway 2 is used for "$\neg x_2$", in which the phosphorylation of MBS and dephosphorylation of MLC are activated, respectively. Pathway 1 consists of sub-pathway 1.1:

$$\text{(p-MBS)--Q + MBS phosphatase + ADP} \rightarrow \text{(MBS\^{}-L)--Q + ATP,}$$

where MBS--Q is rejected by this pathway, and sub-pathway 1.2:

$$\text{(MBS\^{}-L)--Q + p + ATP + L-kinase + MBS kinase} \rightarrow \text{((p-MBS)\^{}-(L-p))--Q + ADP.}$$

Pathway 2 consists of sub-pathway 2.1:

$$\text{(MLC)--Q + MLC kinase + ATP} \rightarrow \text{(p-MLC)\^{}-L)--Q + ADP,}$$

where (p-MLC)--Q will be rejected by this pathway, and sub-pathway 2.2:

((p-MLC)^-L)--Q + p + ATP + L-kinase + MLC phosphatase → (MLC^-(L-p))--Q + ADP.

Consequently, the molecular complexes we can get from pathway 1 are: ((p-MBS)^-(L-p)-Q), i.e., ((p-MBS)^-(L-p)--MLC) and ((p-MBS)^-(L-p)--(MLC-p); from pathway 2, we can get: (MLC^-(L-p)--Q), i.e., ((MLC^-(L-p)-MBS and (MLC^-(L-p)--MBS-p). Finally, we find the solution as: MBS--(MLC-p), ((p-MBS)--MLC) and (p-MBS)--MLC-p, where the L-related molecules are omitted.

2.3 Brief Quantitative Analysis

In the example of the (2, 1) sub-unit, clause selection is regulated by the Rho-MBS-MLC pathways with labeling. Let reactants A_ρ, B_ρ, C_ρ, D_ρ correspond to the "concentration vs. time" curves of the pathways that "process" the objects of (p-MBS)--MLC, MBS-MLC, MBS--(MLC-p) and (p-MBS)--MLC, respectively. These pathway-based computing processes are shown in Fig. 1 where A_ρ, B_ρ, C_ρ, D_ρ are given in series 1, 3, 5 and 2, respectively, and series 4 denotes the related threshold. Through pathways 1 and 2, the reactants A_ρ, B_ρ, C_ρ, D_ρ behave differently. The concentration of A_ρ decreases when the corresponding reaction is carried out and the concentration of D_ρ decreases in a similar way with the related reaction. However, the changes in A_ρ are faster than those in D_ρ.The concentration of B_ρ is consumed slowly with obvious delays in the phase compared with the two above-mentioned signaling processes. This is because the dephosphorylation of MLC and phosphorylation of MBS have been carried out with different dynamical features (the ratio is set to the order of 1:3 in simulation) simultaneously with a time delay and discount in quantity. C_ρ remains constant owing to the fact that it is not involved in the reactions caused by the above-mentioned pathways. In other words, the performance of the pathways' selection that we can observe from the principal data is satisfactory for our analysis from the viewpoint of physical chemistry, so we can use the current technical terms of detection for running, controlling and "reading out" the temporal processes of kinase computing. Although its product has delays and varies in the range of amplitude, synchronization can be guaranteed. "Concentration vs. time" curves are the results of the two crucial stages of label selection and label detection. Here, we can confirm that the features of the corresponding processes given in Fig. 1 are biological faithfully at a certain degree of consistency with the evidence reported in [7].

2.4 Discussion

The stability of differential concentration vs. time curves can be obtained in terms of the feedback mechanism of signaling in cells, which has been discovered by biologists [9]. The cost (i.e., difficulty) of designing pathways for kinase computing mainly depends on activating the target molecules and related regulated pathways for immnofluorescence analysis, which is a practical and efficient technical tool in laboratories. Scalability can be reasonably expected, in a condition sense, owing to

the fact that at least 2000 kinases and about 1000 phosphatases have been discovered in molecular biology [8]; Furthermore, these can be employed for word design and programming by a "kinase computer". One of the most important tasks in regulatory mechanisms of engineered phosphorylation-dephosphorylation processes, which are repeated in cycles, is the decoupling of the phosphorylation-dephosphorylation processes and control of the directed signaling molecules that have a switch-like function in the underlying pathways. Controlled synchronization lays the foundation for all units in kinase computing. The corresponding control strategy and techniques can be developed to guarantee the stability of the entire molecular computation process through well-designed regulation schemes. With these regulation schemes we designed in simulation, we can observe that the time complexity is O (m) and the "regulation-space (control-space)" complexity is O (m $\times$ n) respectively when the "kinase computing" algorithm is applied to solving the 3-SAT problem.

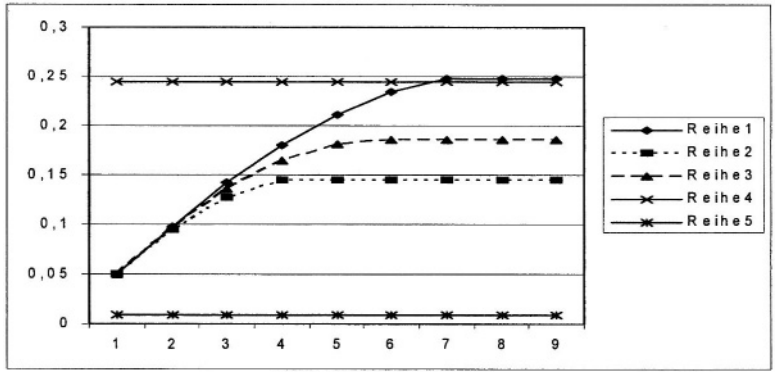


Fig. 1. "Concentration vs. time" curves

3 Conclusion

We have studied a new way to build a cell-based computing system using signaling pathways. The simulation for the related computing model is consistent with the benchwork evidence in biology to a certain degree. It is a novel way to explore cell-scale molecular assembly technologies toward the fabrication of engineered signaling pathways embedded with signal transduction. This development forms the basis for further study on practical scalability among biomolecular computers. Other indirect possible features derived from the core mechanism of the signaling pathways presented here include reliability, robustness and efficiency. Such a mechanism will be helpful in facing the challenges posed by biomolecular computing (e.g., [10]).

Acknowledgement

The authors are thankful to Prof. Kozo Kaibuchi, Dr. Shinya Kuroda and Dr. Mutsuki Amano for their help and suggestions on signal transduction of cell biology, Prof. John H. Reif and Prof. Junghuei Chen for their academic help in discussions and

business help at DNA9, and the anonymous referees and Mr. Darrin Bentivegna for their helpful suggestions to improve this manuscript. This research was conducted as part of "Research on Human Communication" with funding from the Telecommunications Advancement Organization of Japan.

References

[1] Landweber, L.F., Kari, L.: The evolution of cellular computing: nature's solution to a computational problem. BioSystem 52 (1999) 3–13

[2] Ehrenfeucht, A., Harju, T., Petre, I., Rozenberg, G.: Patterns of micronuclear genes in ciliates. In: Jonoska, N., Seeman, N.C. (eds.): DNA7. Lecture Notes in Computer Science, Vol. 2340. Springer-Verlag, Berlin Heidelberg New York (2002) 279–289

[3] Weiss, R., Knight, T.F. Jr.: Engineered communications for microbial robotics. In: Condon, A., Rozenberg, G. (eds.): DNA6. Lecture Notes in Computer Science, Vol. 2045. Springer-Verlag, Berlin Heidelberg New York (2000) 1–16

[4] Păun, G.: From cells to computers: computing with membranes (P systems). BioSystem 59(2001)139–158

[5] Regev, A., Shapiro, E.: Cellular abstractions: computation. Nature 419 (2001) 343

[6] Liu, J.-Q., Shimohara, K.: Kinase Computing in GTPases: analysis and simulation. IPSJ 2001 (2001) 29–32 (in Japanese)

[7] Kaibuchi, K., Kuroda, S., Amano, M.: Regulation of the cytoskeleton and cell adhesion by the Rho family GTPases in mammalian cells. Annu. Rev. Biochem 68 (1999) 459–485

[8] Helmriech, E.J.M.: The Biochemistry of Cell Signaling. Oxford University Press, Oxford, New York (2001)

[9] Freeman, M.: Feedback control of intercellular signaling in development. Nature 408 (2000) 313–319

[10] Reif, J.H.: Successes and Challenges. Science 296 (2002) 478–479

Conformational Addressing Using the Hairpin Structure of Single-Strand DNA

Atsushi Kameda[1], Masahito Yamamoto[2], Hiroki Uejima[3], Masami Hagiya[3], Kensaku Sakamoto[4], and Azuma Ohuchi[2]

[1] Japan Science and Technology Cooperation (JST) Honmachi 4-1-8 Kawaguchi 332-0012, Japan
kameda@dna-comp.org

[2] Division of Systems and Information Engineering, Graduate School of Engineering Hokkaido University, North 13, West 8, Kita-ku, Sapporo 060-8628, Japan
{masahito, ohuchi}@dna-comp.org

[3] 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, uejima}@is.s.u-tokyo.ac.jp

[4] Department of Biophysics and Biochemistry, Graduate School of Science, University of Tokyo 2-11-16 Yayoi, Bunkyo-ku, Tokyo 113-0032, Japan
sakamoto@biochem.s.u-tokyo.ac.jp

Abstract. In this paper, we demonstrate through a chemistry experiment that conformational addressing can be achieved using the hairpin structure of a DNA molecule. The hairpin structure made by single-strand DNA (ssDNA) self-hybridization is made into the address part of conformational addressing, and it is assumed that the memory is read by opening this hairpin. The hairpin is continuously arranged in order to divide the address by class. Reading a sub-address requires an appropriate input oligomer to be added, when the preceding hairpin has been opened. We investigated, through the chemistry experiment, whether it would be possible to open the hairpin by the addition of an input oligomer into a solution that contained a hairpin-formed ssDNA.

1 Introduction

The Adleman study [1] spawned the DNA computing research field, which calculates using a DNA molecule. Recently, the direction of DNA computing research has shifted from problem solving to much broader applications using DNA molecule. Various nanomachines made of DNA molecules have been implemented. Yurke [4] proposed a molecular system called molecular tweezers, which has two state by changing its DNA secondary structures. Yan [5] applied "fuelling" established by the above works to robustly control state transition between JX2 and PX tiles of DNA.

To implement multi-state machines by molecules is a basis of constructing more general nanoscale machines. In this study we investigate conformational

J. Chen and J. Reif (Eds.): DNA9, LNCS 2943, pp. 219–224, 2004.

change of hairpin structure of ssDNA through chemical experiments. We aims at constructing a multi-state molecular machine which makes sequential state transitions by several inputs. Our model retains (i) simple system that made by ssDNA (ii) using conformational change of the hairpin structure.

2 Theoretical Background

2.1 Hairpin Structure of ssDNA

The hairpin structure is obtained combining the ssDNA in self-hybridization, if a complementary portion exists in its sequence. If hairpin can open selectively in the solution, this conformational change of the hairpin can be used as the address portion of a memory by transposing the state in which the hairpin opened and closed. The sticky end sequence is added before the hairpin of the ssDNA molecule to make toehold for opening the hairpin. This ssDNA named the "Hairpin template" (Fig. 1 (A)). We created a DNA oligomer that has a complementary sequence to the sticky end of hairpin temaplate and the stem portion following the sticky end (red line in hairpin template). This is called the "Input oligomer". If two kinds of these DNA molecules are mixed into the solution, the following states can occur ;Fisrt the sticky end of Hairpin template (green line) and its complementary sequence of Input oligomer (green dotted) combine (Fig. 1 (B)). Next, the stem sequence of hairpin (red line in Hairpin template) and its complementary sequence of Input oligomer (red dotted in Input oligomer) combine, so hairpin structure can open (Fig. 1 (C)). Finally, all the domains of an Input oligomer form double-strands with the hairpin template, and the stem portion of the hairpin leaves (Fig. 1 (D)). When a structure prediction of DNA was performed based on the research of the structure prediction of RNA [2], it appeared possible to open the hairpin structure with this method [3]. If opening the hairpin is controllable by this method, this conformational change is able to use for addressing in DNA memory.

The hairpin createrd in the ssDNA through self-hybridization is inserted into the address part of the conformational addressing. The hairpin is continuously arranged in order to divide an address by class. Each hairpin represents a sub-address, and reading a sub-address requires an appropriate Input oligomer to be added into the solution after the preceding hairpin has been opened.

2.2 Reading Specific Data from Conformational Addressing Memory

Reading memory using hairpin confromational addressing (Fig. 2) is accomplished, first, by putting hairpins in continuous order and building a sub-address. This is considered an address block (Fig. 2 (A)). A data block is then added, and this data block corresponds to each address block. Such a ssDNA molecule is prepared in large quantities into a solution. At first, in all the ssDNA molecules, all the hairpins are closed, and only the hairpin at the very end of the 5' end side

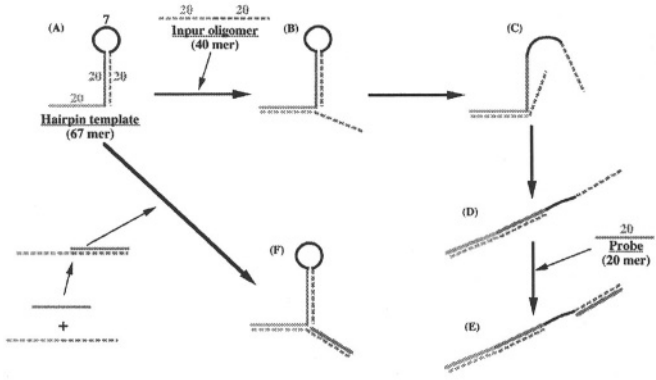


Fig. 1. Opening of the hairpin structure of ssDNA by the addition of DNA oligomer

(red character in Fig. 2 (A)) can be accessed. This can be opened by the adding corresponding Input oligomer to the solution. When the hairpin has opened, it becomes possible to access the next hairpin because of exposing toehold that is needed to open next hairpin; One side of the stem portion of the opened hairpin has combined with the Input oligomer. However, one of the two is in the single-strand state. This is determined to be the sticky site that is required to open the next hairpin (Fig. 2 (B)). Following this step, the Input oligomer is inserted for the hairpin located in a line with the address block corresponding to the data to be read.

Only the target ssDNA molecule can open the hairpin of all the sub-addresses by it. Thus, it becomes possible to access the target data block (Fig. 2 (C)).

3 Verification of Realizationizing by Chemistry Experiment

We verified whether conformational addressing using the hairpin structure of ssDNA could be realized through a chemistry experiment.

3.1 Materials and methods

We prepared “hairpin temaplte”, “input oligomer” and “probe” (described in Fig. 1). These ssDNAs length were determined from a stabilized and existing hairpin [3]. The fact that it is necessary to put this hairpin in a successive order of great number as a prospective target is taken into consideration from the sequence design stage. The sequence designing method that took into consideration the effectiveness with which the hairpin is opened by the input oligomer was adopted. In order to check whether the hairpin is open, a probe combined with the stem portion of the hairpin contrary to the combined sequence with the input oligomer was prepared. The sequences of these ssDNAs are described

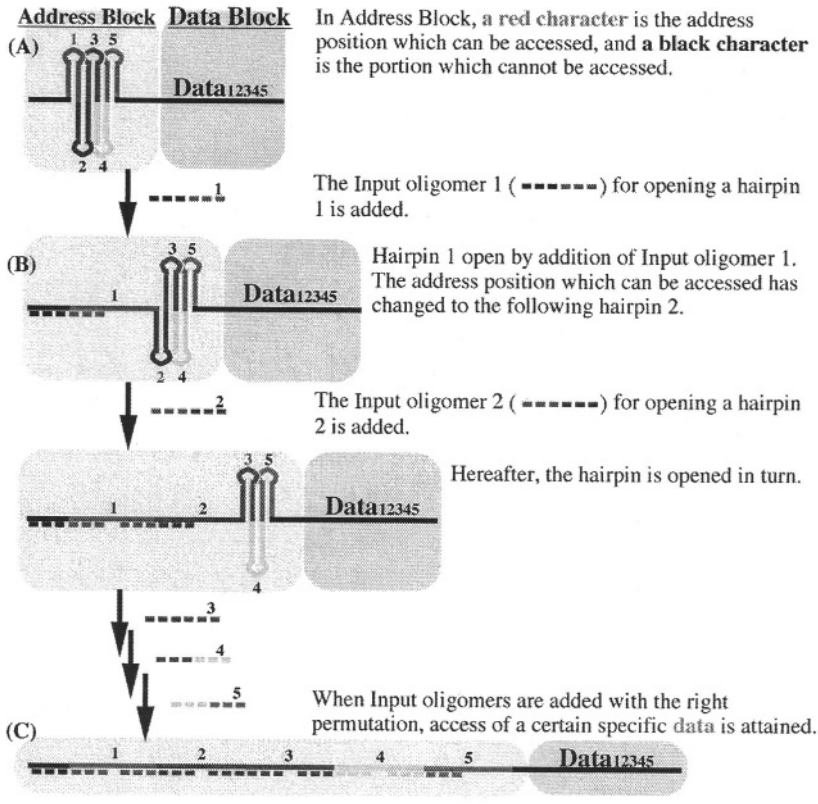


Fig. 2. Conformational addressing using the hairpin structure of single-strand DNA

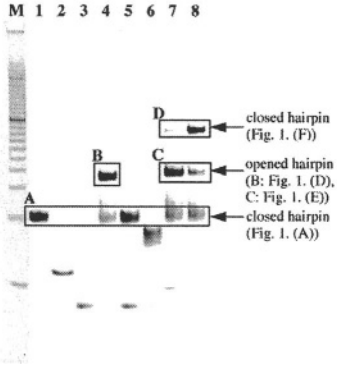
in Table 1. The solution buffer were contained 10 mM TrisHCl (pH 8.0) and 2.5 mM $MgSO_4$. Each DNA oligomer concentration was 1 μM in a combination solution, mixed various combination (Fig. 3 Lane data). We performed 10 % non-denaturing polyacrylamide gel electrophoresis (PAGE).

3.2 Results

The band of the hairpin template that forms the hairpin was the A in Fig. 3. In the combination of the hairpin template and the probe (Fig. 3 lane 5), the probe was uncombinable with the hairpin template because the stem sequence needed for this combination had formed a hairpin. In the combination of the hairpin template and input oligomer, new band appeared in the B (Fig. 3 lane 4). This band showed the DNA duplex of the hairpin template and input oligomer. However, it could not be determined by this result which state of (B) or (D) in Fig. 1 was taken. From lane 7 and 8 in Fig. 3, it is considerable that band C is the state of (E) in Fig. 1, and band D is the state of (F) in Fig. 1. All the combination are react at room temperature.

Table 1. Designed sequences of DNA oligomer for the opening test of the hairpin structure of ssDNA

Name	Sequence
Hairpin template	5'–CATCCTTAGGTCTGGCATGCGGGATGCCATCTCA CACTTCACATAAAGAAGTGTGAGATGGCATCCC–3'
Input oligomer	5'–GAAGTGTGAGATGGCATCCCGCATGCCAGACCTAAGGATG–3'
Probe	5'–GGGATGCCATCTCACACTTC–3'



PAGE condition: 10% non-denaturing PAGE, 200 V const, 35 minute. The DNA applied to the PAGE were visualized by SYBR Gold staining (Molecular Probes, Inc.).
Lane data: (each lane were applied 2 μl DNA mixed solution which contain 1 μM each ssDNAs)
1: Hairpin template 2: Input oligomer 3: Probe
4: Hairpin template + Input oligomer
5: Hairpin template + Probe
6: Input oligomer + Probe
7: (Hairpin template + Input oligomer) after 1 minute interval, added Probe
8: (Input oligomer + Probe) after 1 minute interval, added Hairpin template

Fig. 3. Opening test of the hairpin structure of ssDNA with 10% non-denaturing PAGE

3.3 Discussion

Experimental results proved that our method can open the hairpin of the Hairpin template at room temperature by the addition of the Input oligomer into the solution. This verifies the basic operation of conformational addressing using the hairpin structure of ssDNA. In order to divide an address by class, a continuous hairpin structure needs to be built. A method of opening that maintains the sequentiality of the continuous hairpin is also required. At present, it is possible to construct four continuous hairpin structures at most, and satisfactory results for opening correctly have been obtained (data not shown). In order to check whether the address divided by class can be read correctly, more experiments must be conducted.

4 Concluding Remarks

In this paper, the construction of conformational addressing using the hairpin structure of ssDNA was proposed. Experimental results probed that our method

of conformational addressing can be achieved. Further verification experiments will be conducted.

References

[1] L. Adleman: Molecular Computation of Solutions to Combinatorial Problems, Science, vol. 266, pp. 1021-1024, 1994.
[2] M. Zuker and P. Stiegler: Optimal Computer Folding of Large RNA Sequences Using Thermodynamics and Auxiliary Information, Nucleic Acids Research 9, pp. 133-148, 1981.
[3] Hiroki Uejima and Masami Hagiya: Secondary Strucure Design of Multi-state DNA Machine Based on Sequential Structure Transtions, submitted to the Ninth International Meeting on DNA Based Computers (DNA9), 2003.
[4] B. Yurke et. al:DNA-fuelled molecular machine made of DNA, Nature vol. 406, pp. 605-608, 2000.
[5] Hao Yan, Xiaoping Zhang, Zhiyong Shen, Nadrian C. Seeman :A robust DNA mechanical device controlled by hybridization topology, Nature vol. 415, pp. 62-65, 2002.

Author Index