LNCS 6518

Yasubumi Sakakibara
Yongli Mi (Eds.)

DNA Computing and Molecular Programming

16th International Conference, DNA 16
Hong Kong, China, June 2010
Revised Selected Papers

Springer

Yasubumi Sakakibara   Yongli Mi (Eds.)

DNA Computing and Molecular Programming

16th International Conference, DNA 16
Hong Kong, China, June 14-17, 2010
Revised Selected Papers

Springer

Volume Editors

Yasubumi Sakakibara
Keio University, Department of Biosciences and Informatics
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522, Japan
E-mail: yasu@bio.keio.ac.jp

Yongli Mi
Hong Kong University of Science and Technology
Department of Chemical and Biomolecular Engineering
Clear Water Bay, Kowloon, Hong Kong, China
E-mail: keymix@ust.hk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-18304-1 e-ISBN 978-3-642-18305-8
DOI 10.1007/978-3-642-18305-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2010942453

CR Subject Classification (1998): F.1, F.2.2, J.3, E.1, I.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The 16th International Meeting on DNA Computing and Molecular Programming was held during June 14–17, 2010 in Hong Kong University of Science and Technology, Hong Kong. This conference series serves as one of the many task forces aimed at achieving a mission to establish biomolecular computing in life science. Through the efforts of many scientists in the past 16 years, accomplishments from DNA computing to programmed construction of 1D, 2D, 3D DNA scaffolds, and DNA origami have been achieved and reported. Participants of this conference carry out well-funded research projects in DNA-controlled drug delivery systems, DNA nanomachines, DNA-induced information storage devices, DNA scaffolds and patterning, etc. Scientists, engineers, and students meet at this time every year to present new results and project foreseeable goals for the future.

The scientific program included three tutorials, six invited lectures, 23 oral presentations, and 31 posters. The topics were well balanced between theoretical and experimental work. The meeting began with tutorial talks by John Reif, Chengde Mao, and Brian Wolfe. During the meeting, a number of excellent keynote speakers gave an up-to-date overview of different aspects of DNA computing and biochemical information processing. We express our appreciation to Ned Seeman, Akira Suyama, Christina Smolke, Satoshi Kobayashi, David Soloveichik, and Erik Demaine for their excellent keynote talks. Thanks are also given to all the authors of the oral presentations and posters. Their efforts made this meeting possible.

In total, the meeting was attended by 92 researchers from 13 countries: Canada, China, Finland, Germany, Ireland, Israel, Italy, Japan, South Korea, Singapore, Spain, UK, USA. The DNA16 Program Committee received a total number of 59 submissions, of which 23 were presented orally. This proceedings volume contains improved versions of 16 papers selected from these oral contributions.

The editors would like to thank the members of the Program Committee and the reviewers for all their hard work reviewing papers and providing constructive comments to authors. We give thanks to Natasha Jonaska, the Chair of the Steering Committee, Llyod Smith, the Co-chair of the Program Committee, and Shihua Zhang for maintaining the conference website and facilitating communications.

October 2010

Yasubumi Sakakibara
Yongli Mi

Organization

DNA16 was organized by Hong Kong University of Science and Technology.

Program Committee

Organizing Committee

Steering Committee

External Reviewers

Sponsoring Institutions

Institute for Advanced Study of the Hong Kong University of Science and Technology
School of Engineering and the Department of Chemical and Biomolecular Engineering of the Hong Kong University of Science and Technology
Lee Hysan Foundation

Table of Contents

Improving Efficiency of 3-SAT-Solving Tile Systems

Yuriy Brun

University of Washington, Seattle, WA 98195-2350, USA
brun@cs.washington.edu

Abstract. The tile assembly model has allowed the study of the nature's process of self-assembly and the development of self-assembling systems for solving complex computational problems. Research into this model has led to progress in two distinct classes of computational systems: Internet-sized distributed computation, such as software architectures for computational grids, and molecular computation, such as DNA computing. The design of large complex tile systems that emulate Turing machines has shown that the tile assembly model is Turing universal, while the design of small tile systems that implement simple algorithms has shown that tile assembly can be used to build private, fault-tolerant, and scalable distributed software systems and robust molecular machines. However, in order for these types of systems to compete with traditional computing devices, we must demonstrate that fairly simple tile systems can implement complex and intricate algorithms for important problems. The state of the art, however, requires vastly complex tile systems with large tile sets to implement such algorithms.

In this paper, I present $\mathbb{S}_{FS}$, a tile system that decides 3-SAT by creating $O^\star(1.8393^n)$ nondeterministic assemblies in parallel, while the previous best known solution requires $\Theta(2^n)$ such assemblies. In some sense, this tile system follows the most complex algorithm implemented using tiles to date. I analyze that the number of required parallel assemblies is $O^\star(1.8393^n)$, that the size of the system's tileset is $147 = \Theta(1)$, and that the assembly time is nondeterministic linear in the size of the input. This work directly improves the time and space complexities of tile-inspired computational-grid architectures and bridges theory and today's experimental limitations of DNA computing.

1 Introduction

Self-assembly is a process by which simple objects in nature combine and coordinate to form complex objects. For computer scientists, it is interesting to study self-assembly from a computational point of view as self-assembling systems have been shown capable of computing functions [2,19], assembling complex shapes [15,17], and guiding distributed robotics systems [1,12].

The tile assembly model [20,15] is a formal mathematical model that allows studying the time and space complexities of self-assembling systems. Winfree showed that the tile assembly model is Turing universal [19] by demonstrating that tile systems can emulate Turning machines. Adleman has identified two important measures of tile systems: assembly time and tileset size; in some ways, these measures are analogous to the time and space complexities of traditional computer programs [2].

Study of tile systems has led to two types of computational systems: Internet-sized distributed grids [8] and molecular computers [2]. Both types benefit from efficient tile systems with small tilesets: the speed of computational grids is proportional to the number of tile types [8] and the state of the art in DNA computation is systems with no more than tens of distinct tile types [3,14]. Winfree's universal tile systems are in some sense inefficient and require thousands of distinct tile types [19]. Lagoudakis et al. [11] presented a tile system that solves 3-SAT, though their best solution required $\Theta(n^2)$ distinct tile types and $\Theta(2^n)$ distinct nondeterministic assemblies to solve an n-variable problem, resulting in over 10^5 distinct tile types and 10^{15} distinct assemblies necessary to solve a 50-variable problem. I have begun the work of reducing the complexity by designing tile systems that solve complex computational problems using relatively small tilesets (e.g., adding using 8 distinct tile types [4], multiplying using 28 [4], factoring integers nondeterministically using 50 [5], and solving two NP-complete problems nondeterministically, 3-SAT using 64 [7] and *SubsetSum* using 49 [6]). However, thus far, existing tile systems implement only the most naïve, simple, and inefficient algorithms. For example, today's best known NP-complete problem-solving tile systems require $\Theta\left(2^n\right)$ distinct assemblies for an input of size n. While we do not know of polynomial-time algorithms to solve such problems, we do know of exponential-time algorithms with a base smaller than 2 [21]. Here, I present a tile system that implements a somewhat complex known algorithm for solving 3-SAT using only $O^\star(1.8393^n)$ distinct assemblies (where the $O^\star$ notation hides constant and polynomial factors). This system uses 147 distinct tile types and demonstrates that complex algorithms can be implemented using tiles in a systematic manner, (1) directly improving the time and space complexities of tile-inspired computational-grid architectures [8] and (2) bridging theory and today's experimental limitations of DNA computing [2,3].

3-SAT is a well known NP-complete problem of deciding whether a 3CNF Boolean formula is satisfiable. The naïve algorithms for solving 3-SAT explore the $\Theta\left(2^n\right)$ distinct assignments, for formulae with n distinct variables, checking if any one of them satisfies the formula. While we are unaware of subexponential-time algorithms to solve NP-complete problems, there are algorithms that perform in exponential time but with a base smaller than 2. Woeginger [21] provides a fairly complete survey of more-efficient exponential-time algorithms for 3-SAT, one of which I employ here. For my discussion, I define the $O^\star$ notation, which is similar to the O notation but ignores both constant and polynomial factors. Thus I will say $O^\star(m(x))$ for a complexity of the form $O(m(x) \cdot poly(x))$. The justification for this notation is that the exponential growth of $m(x)$ will dominate all polynomial factors for large x. For example, if f is a function such that $f(x) = O\left(1.4142^x x^4\right)$, then I write $f(x) = O^\star(1.4142^x)$. Note that the exponential term dominates and one could say $f(x) = O(1.4143^x)$ and forgo the

$O^\star$ notation altogether; however, that would not most accurately describe the functions.

While the naïve algorithms explore each of the possible 2^n truth assignments to the n variables, a more intricate algorithm can explore a subset of those assignments by noting the following fact: if the Boolean formula contains the clause $(x_1 \vee \neg x_2 \vee x_3)$, then the algorithm need not explore any of the $2^{(n-3)}$ assignments with $x_1 = x_3 = \mathit{FALSE}$ and $x_2 = \mathit{TRUE}$ because this clause would not be satisfied by any of those assignments. Instead, the algorithm explores only assignment with (1) $x_1 = \mathit{TRUE}$, or (2) $x_1 = x_2 = \mathit{FALSE}$, or (3) $x_1 = \mathit{FALSE}$, $x_2 = \mathit{TRUE}$, and $x_3 = \mathit{TRUE}$. Thus, deciding an n-variable m-clause Boolean formula can be done by recursively deciding three Boolean formulae: each with one fewer clause and with one, two, and three fewer variables, respectively. Thus if $T(n, m)$ denotes the time necessary to decide an n-variable m-clause Boolean formula, then $T(n, m) = O(1) + T(n-1, m-1) + T(n-2, m-1) + T(n-3, m-1)$. This recurrence has the closed form solution $T(n, m) = O^\star(1.8393^n)$ [21]. By examining the branching step, it is possible to improve the algorithm further to an $O^\star(1.6181^n)$ algorithm [13]. Using quantitative analysis of the number of resulting 2-clauses from such branching improves the time complexity to $O^\star(1.5783^n)$ [16]. The champion algorithm using this technique achieves a time complexity of $O^\star(1.4963^n)$ [9,10], and other techniques result in even faster algorithms [21]. It is not my goal to explore the fastest such algorithm here, but rather to demonstrate that it is possible to implement one such complex algorithm using a tile system with a small tileset. I will thus concentrate on developing a tile system that follows the $O^\star(1.8393^n)$ algorithm, and argue that since the other algorithms are similar, it is possible to design tile systems for those algorithms as well.

2 Tile Assembly Model

The tile assembly model [20,15] is a formal model of crystal growth. It was designed to model self-assembly of molecules such as DNA. It is an extension of a model proposed by Wang [18]. The model was fully defined in [15], and the definitions I use are similar to those. Full formal definitions can be found in [7].

Intuitively, the model has *tiles*, or squares, that stick or do not stick together based on various *binding domains* on their four sides. Each tile is a four tuple of binding domains, one on each (north, east, south, and west) of its sides. The special $\mathit{empty} = \langle \mathit{null}, \mathit{null}, \mathit{null}, \mathit{null} \rangle$ tile denotes an empty position. The four binding domains, elements of a finite alphabet Σ, define the type of the tile. The strength of the binding domains are defined by the *strength function* $g\colon \Sigma \times \Sigma \to \mathbb{N}$. A mapping from positions on a 2-D grid to tiles is called a *configuration*, and a tile may *attach* in empty positions on the grid if the total strength of all the binding domains on that tile that match its neighbors exceeds the current *temperature*. Finally, a *tile system* $\mathbb{S}$ is a triple $\langle T, g, \tau \rangle$, where T is a finite set of tiles, g is a strength function, and $\tau \in \mathbb{N} = \mathbb{Z}_{\geq 0}$ is the temperature.

Starting from a *seed configuration* S, tiles may attach to form new configurations. If that process terminates, the resulting configuration is said to be *final.* At some times, it may be possible for more than one tile to attach at a given position, or there may be more than one position where a tile can attach. If for all sequences of tile attachments, all possible final configurations are identical, then $\mathbb{S}$ is said to produce a *unique* final configuration on S. The *assembly time* of the system is the minimal number of steps it takes to build a final configuration, assuming maximum parallelism.

In solving NP-complete problems, it is important to compute a particular subset of functions: the characteristic functions of subsets of the natural numbers. A characteristic function of a set has value 1 on arguments that are elements of that set and value 0 on arguments that are not elements of that set. Typically, in computer science, programs and systems that compute such functions are said to decide the set. Since for all constants $n \in \mathbb{N}$, the cardinalities of $\mathbb{N}^n$ and $\mathbb{N}$ are the same, one can encode an element of $\mathbb{N}^n$ as an element of $\mathbb{N}$. Thus it makes sense to talk about deciding subsets of $\mathbb{N}^n$. Let $\Omega \subseteq \mathbb{N}^m$ be a set. A tile system $\mathbb{S} = \langle T, g, \tau \rangle$ *nondeterministically decides* Ω with identifier tile $r \in T$ iff for all $\boldsymbol{a} \in \mathbb{N}^m$, there exists a seed configuration S that encodes $\boldsymbol{a}$ and for all final configurations F that $\mathbb{S}$ produces on S, $r \in F(\mathbb{Z}^2)$ iff $\boldsymbol{a} \in \Omega$, and there exists at least one final configuration F with r attached. In other words, the *identifier* tile r attaches to one or more of the nondeterministic executions iff the seed encodes an element of Ω. I call the set of tiles used to encode the input Γ.

I have given informal definitions to assist the reader in understanding the system I discuss in this paper and I refer the reader to [7] for more formal definitions.

3 Solving 3-SAT Efficiently with Tiles

Implementing algorithms in the tile assembly model is not unlike implementing algorithms using Turing machines, or programming using a low-level language, such as assembly. The complexity of that process has led to only simple algorithms implemented into tile systems. Here, I propose the tile system $\mathbb{S}_{FS}$ (FS stands for "fast satisfiability") that implements the $O^\star(1.8393^n)$ algorithm for solving 3-SAT. The algorithm's running time implies that $\mathbb{S}_{FS}$ will create $O^\star(1.8393^n)$ distinct assemblies to decide an n-variable formula.

$\mathbb{S}_{FS}$ is a combination of several subsystems, each with a distinct job. Figure 1 shows the general placement of the distinct subsystems on a 2-D grid. The overall system will construct a right triangle, starting from region I, which encodes a Boolean formula ϕ. Region II will examine the eastmost clause of ϕ and determine which literals have not been assigned a value (at the start of the computation, there will always be 3 unassigned literals in each clause of a 3-SAT formula, but as the algorithm makes assignment decisions, clauses may have fewer such literals). Region III will make the nondeterministic decision on what

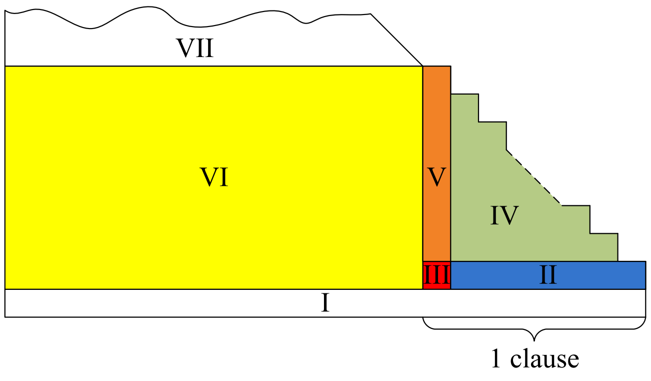


Fig. 1. A schematic of the seven regions $\mathbb{S}_{FS}$ will use to decide 3-*SAT*.

assignments to make regarding the unassigned literals in the eastmost clause. Region IV will prepare the literals of the eastmost clause to be assigned by the decision made in region III, and region V will make those assignments. Region VI will simplify the rest of ϕ based on those assignments. At the top of region VI, the simplified ϕ, with one fewer clause, will emerge to serve as the input (like region I) for the remainder of the computation in region VII. That is, $\mathbb{S}_{FS}$ will operate recursively in Region VII on the simplified ϕ.

The rest of this section demonstrates that $\mathbb{S}_{FS}$ decides 3-*SAT* requiring only $O^\star(1.8393^n)$ distinct assemblies. Due to space limitations, I am unable to include the appropriate details and proofs here.

3.1 Notations and Definitions

Let ϕ be a Boolean formula. Let n be the number of distinct variables and m be the number of clauses in ϕ. A literal over a variable x is an element of $\{x, \neg x\}$. As is common, I assume that no clause of ϕ contains more than one literal over the same variable.

For all m, for all n, for all n-variable, m-clause 3CNF Boolean formulae ϕ, ϕ is a *general* 3CNF Boolean formula iff each of the three literals of each clause either is identically a variable, is the negation of a variable, or is represented by *TRUE* or *FALSE* and no pair of literals within each clause are over the same variable.

The tile system $\mathbb{S}_{FS}$ will operate at temperature 2, and will use a fairly straightforward strength function g_{FS} over the set of binding domains Σ_{FS} = {*null*, t, bt, bbt, ft, fft, fbt, bft, T, F, @, @*, 0, 1, 0t, 1t, 0f, 1f, x, ¬x, x*, ¬x*, c, #, 0#, 1#, #f, #t, ::, 0:, 1:, 2:, 3:, 0:*, 1:*, 2:*, 1:1, 1:1*, 2:1, 2:1*, 3:1, 2:2, 2:2*, 3:2, 2:12, 2:12*, 3:12, **, *, |}. For the most part, g_{FS} will match two identical binding domains to 1, and two different binding domains to 0, with a few special wildcard binding domains that will map to 1 even with some unmatching domains. In other words, all attachments are either strength 0 or 1, as described in Figure 2.

	Intuitively, g_{FS} is such that:	Formally, $g_{FS}\colon \Sigma_{FS} \times \Sigma_{FS} \to \{0,1\}$ such that:
0.	*null* does not attach to anything,	for all $\sigma \in \Sigma_{FS}$, $g_{FS}(null, \sigma) = g_{FS}(\sigma, null) = 0$,
1.	every binding domain attaches to itself,	for all $\sigma \in \Sigma_{FS} \setminus \{null\}$, $g_{FS}(\sigma,\sigma) = 1$,
2.	# attaches to 0, 1, x, ¬x, 1f, 1t, 0f, 0t, T, and F,	for all $\sigma \in \{\mathsf{0}, \mathsf{1}, \mathsf{x}, \neg\mathsf{x}, \mathsf{1f}, \mathsf{1t}, \mathsf{0f}, \mathsf{0t}, \mathsf{T}, \mathsf{F}\}$, $g_{FS}(\#, \sigma) = g_{FS}(\sigma, \#) = 1$,
3.	:: attaches to 0:, 1:, 2:, 1:1, 2:1, 2:2, and 2:12,	for all $\sigma \in \{\mathsf{0:}, \mathsf{1:}, \mathsf{2:}, \mathsf{1:1}, \mathsf{2:1}, \mathsf{2:2}, \mathsf{2:12}\}$, $g_{FS}(::, \sigma) = g_{FS}(\sigma, ::) = 1$,
4.	#f attaches to 0f, 1f, and F,	for all $\sigma \in \{\mathsf{0f}, \mathsf{1f}, \mathsf{F}\}$, $g_{FS}(\#\mathsf{f}, \sigma) = g_{FS}(\sigma, \#\mathsf{f}) = 1$,
5.	#t attaches to 0t, 1t, and T,	for all $\sigma \in \{\mathsf{0t}, \mathsf{1t}, \mathsf{T}\}$, $g_{FS}(\#\mathsf{t}, \sigma) = g_{FS}(\sigma, \#\mathsf{t}) = 1$,
6.	0# attaches to 0, 0f, and 0t,	for all $\sigma \in \{\mathsf{0}, \mathsf{0f}, \mathsf{0t}\}$, $g_{FS}(\mathsf{0}\#, \sigma) = g_{FS}(\sigma, \mathsf{0}\#) = 1$,
7.	1# attaches to 1, 1f, and 1t,	for all $\sigma \in \{\mathsf{1}, \mathsf{1f}, \mathsf{1t}\}$, $g_{FS}(\mathsf{1}\#, \sigma) = g_{FS}(\sigma, \mathsf{1}\#) = 1$,
8.	@⋆ attaches to @ and ⋆,	for all $\sigma \in \{@, \star\}$, $g_{FS}(@^\star, \sigma) = g_{FS}(\sigma, @^\star) = 1$,
9.	and no other pairs of binding domains attach.	and for all other pairs $\sigma, \sigma' \in \Sigma_{FS}$, $g_{FS}(\sigma, \sigma') = g_{FS}(\sigma', \sigma) = 0$.

Fig. 2. The strength function g_{FS}.

3.2 Clause Examination (Region II)

In this Section, I define the tile system $\mathbb{S}_{EXAM}$, which will become the part of $\mathbb{S}_{FS}$ that will operate in region II, as denoted in Figure 1. Since $\mathbb{S}_{FS}$ will operate on the first clause to fill in regions II through VI, and then recurse on the remaining simplified formula in region VII, for each clause there is a distinct copy of each region.

The goal of $\mathbb{S}_{EXAM}$ is to examine the first (eastmost) clause in the formula for the number of unassigned literals. Figure 3(a) shows the 37 tiles of T_{EXAM} that perform this examination. I define the function p that maps clauses of general 3CNF Boolean formulae to binding domains. Let c be a clause of a general 3CNF Boolean formula. Then, if c contains the literal *TRUE*, then $p(c) = \mathsf{T}$; otherwise, the value of $p(c)$ is defined by Figure 4. $\mathbb{S}_{EXAM}$ will attach just to the north of an encoding of clause c and will propagate that encoding one row north and make the west binding domain of the westmost tile be the value of $p(c)$.

3.3 Assignment Selection (Region III)

In this Section, I define the tile system $\mathbb{S}_{SELECT}$, which will become the part of $\mathbb{S}_{FS}$ that will operate in region III, as denoted in Figure 1.

The goal of $\mathbb{S}_{SELECT}$ is to nondeterministically select an assignment over the variables of the eastmost clause just as the $O^\star(1.8393^n)$ algorithm would. Thus, if $\mathbb{S}_{EXAM}$ finds that the clause has three unassigned literals, $\mathbb{S}_{SELECT}$ will pick either (1) the first literal to be true, or (2) the first literal to be false and the

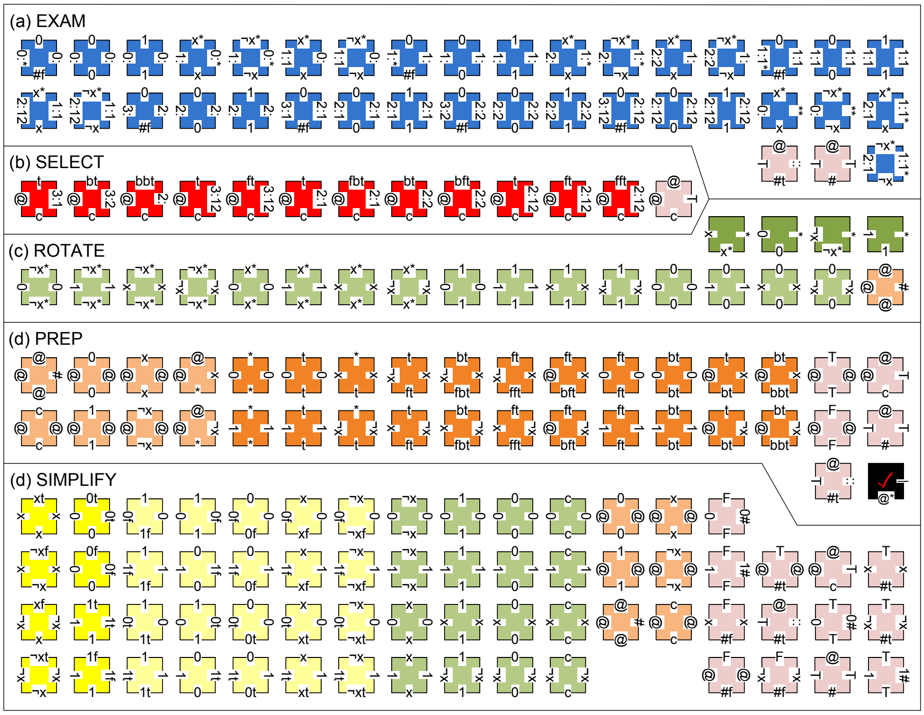


Fig. 3. Tiles of T_{EXAM} (a), T_{SELECT} (b), T_{ROTATE} (c), T_{PREP} (d), and $T_{SIMPLIFY}$ (e). Together, these sets form T_{FS} with 147 distinct tiles.

second to be true, or (3) the first two literals to be false and the third to be true. Alternatively, if $\mathbb{S}_{EXAM}$ finds that the clause has its first literal already assigned and the other two unassigned, $\mathbb{S}_{SELECT}$ will pick to ignore the first literal and either (1) the second literal to be true, or (2) the second literal to be false and the third to be true. And so on.

Figure 3(b) shows the 13 tiles of T_{SELECT} that perform the assignment selection. $\mathbb{S}_{SELECT}$ will attach just to the west of the westmost tile attached by $\mathbb{S}_{EXAM}$ and nondeterministically select one of the possible assignments in $ac(c)$ as that tile's north binding domain.

3.4 Clause Rotation (Region IV)

In this Section, I define the tile system $\mathbb{S}_{ROTATE}$, which will become the part of $\mathbb{S}_{FS}$ that will operate in region IV, as denoted in Figure 1.

The goal of $\mathbb{S}_{ROTATE}$ is to rotate a horizontally positioned clause encoding to be vertically positioned. This rotation later allows $\mathbb{S}_{SIMPLIFY}$ to simplify the formula. $\mathbb{S}_{ROTATE}$ will present the clause encoded in the north binding domains of part of its seed as the west binding domains of the completed right triangle. Figure 3(c) shows the 21 tiles of T_{ROTATE} that perform the rotation.

c's literals			$p(c)$	$ac(c)$
first	second	third		
FALSE	*FALSE*	*FALSE*	3:	{F}
FALSE	*FALSE*	unassigned	3:1	{bbt}
FALSE	unassigned	*FALSE*	3:2	{bt}
FALSE	unassigned	unassigned	3:12	{bt, bft}
unassigned	*FALSE*	*FALSE*	2:	{t}
unassigned	*FALSE*	unassigned	2:1	{t, fbt}
unassigned	unassigned	*FALSE*	2:2	{t, ft}
unassigned	unassigned	unassigned	2:12	{t, ft, fft}

Fig. 4. For every clause c of a general 3CNF Boolean formula, $p(c) = \mathsf{T}$ and $ac(c) = \{@\}$ if c contains the literal *TRUE*, and otherwise, the values of $p(c)$ and $ac(c)$ are defined by this table. The goal of the $\mathbb{S}_{EXAM}$ system will be to produce, on the west side of the westmost tile in region II, the value $p(c)$ of the examined clause, and the goal of the $\mathbb{S}_{SELECT}$ system will be to produce one of the elements of $ac(c)$ on the north side of region III.

3.5 Assignment Preparation (Region V)

In this section, I define the tile system $\mathbb{S}_{PREP}$, which will become the part of $\mathbb{S}_{FS}$ that will operate in region V, as denoted in Figure 1.

The goal of $\mathbb{S}_{PREP}$ is to turn the assignment selected by $\mathbb{S}_{SELECT}$ into up to three literals that evaluate to *TRUE*. In other words, to apply the assignment to the clause. This application of the assignment is the final preparation before $\mathbb{S}_{SIMPLIFY}$ can simplify the formula. $\mathbb{S}_{PREP}$ will present the up to three literals as the west binding domains of the column in which it operates. Figure 3(d) shows the 36 tiles of T_{PREP}.

3.6 Formula Simplification (Region VI)

In this Section, I define the tile system $\mathbb{S}_{SIMPLIFY}$, which will become the part of $\mathbb{S}_{FS}$ that will operate in region VI, as denoted in Figure 1.

The goal of $\mathbb{S}_{SIMPLIFY}$ is to simplify the formula by replacing instances of the up to three literals prepared by $\mathbb{S}_{PREP}$ with *TRUE* and negations of those literals with *FALSE*. $\mathbb{S}_{SIMPLIFY}$ will present an encoding of the simplified formula as the north binding domains of the rectangle in which it operates. Figure 3(e) shows the 63 tiles of $T_{SIMPLIFY}$ that perform the simplification.

3.7 Solving 3-SAT

Thus far, I have described five tile systems: $\mathbb{S}_{EXAM}$, $\mathbb{S}_{SELECT}$, $\mathbb{S}_{ROTATE}$, $\mathbb{S}_{PREP}$, and $\mathbb{S}_{SIMPLIFY}$ that I intend to use to solve 3-*SAT*. These systems will operate in regions II, III, IV, V, and VI in Figure 1, respectively. The tile system $\mathbb{S}_{FS}$ combines these five systems to select a truth assignment for the first (eastmost) clause of a Boolean formula ϕ, simplify the rest of ϕ based on that assignment, and recurse (in region VII) on the simplified ϕ with one fewer clause.

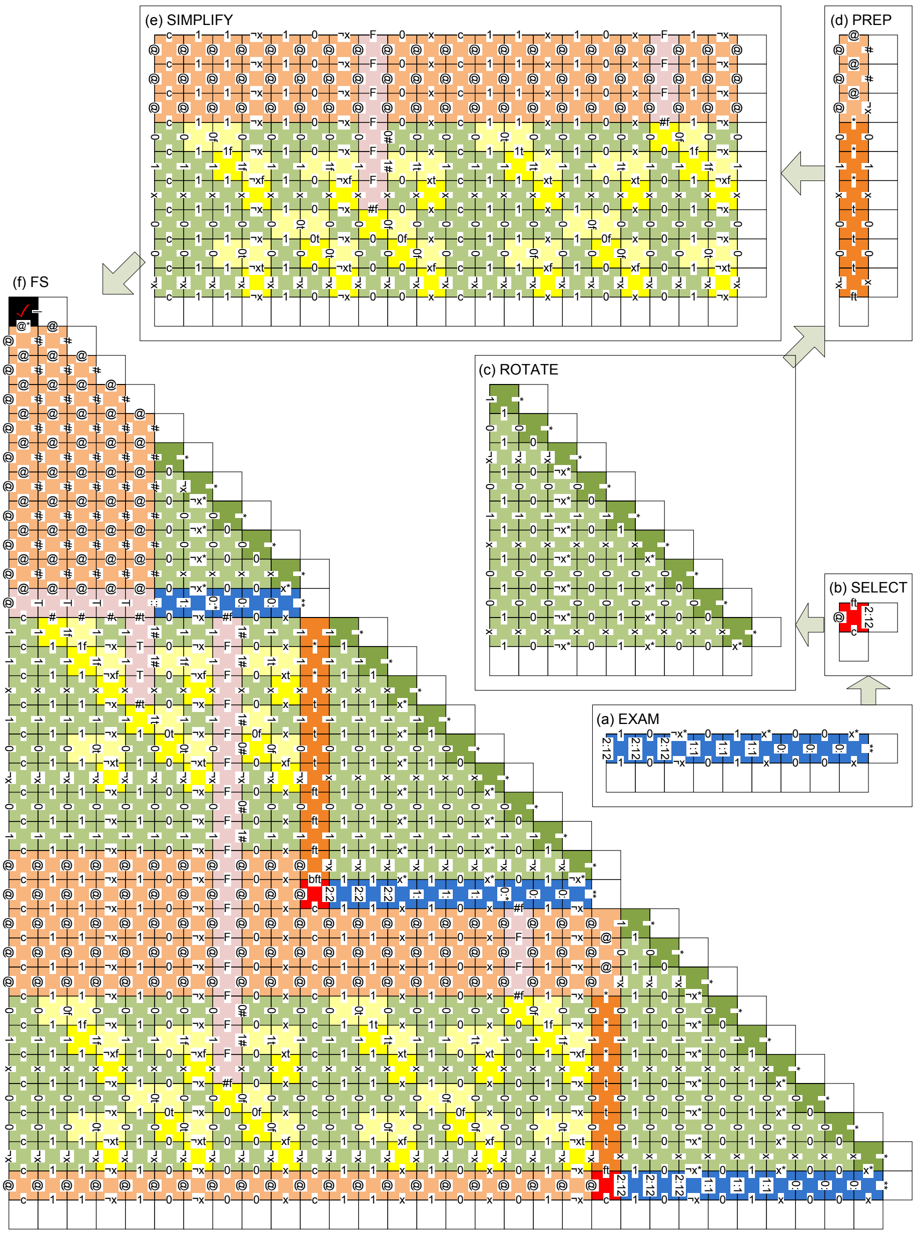


Fig. 5. Example executions of $\mathbb{S}_{EXAM}$ (a), $\mathbb{S}_{SELECT}$ (b), $\mathbb{S}_{ROTATE}$ (c), $\mathbb{S}_{PREP}$ (d), $\mathbb{S}_{SIMPLIFY}$ (e), and $\mathbb{S}_{FS}$ (f). $\mathbb{S}_{FS}$ operates on the Boolean formula $\phi = (\neg x_3 \vee \neg x_2 \vee x_0) \wedge (x_3 \vee x_2 \vee \neg x_1) \wedge (\neg x_2 \vee x_1 \vee x_0)$. The clear tiles are parts of the seed and the shaded tiles are computational. This $\mathbb{S}_{FS}$ execution nondeterministically selects the assignment $x_0 = \mathit{FALSE}$, $x_1 = \mathit{TRUE}$, $x_2 = \mathit{FALSE}$, and $x_3 = \mathit{TRUE}$; because that assignment satisfies ϕ, the black ✓ tile attaches in the northwest corner.

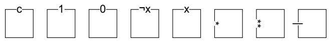


Fig. 6. The 8 tiles of Γ_{FS} used to encode inputs to $\mathbb{S}_{FS}$.

$\mathbb{S}_{FS}$ will nondeterministically create only $O^\star(1.8393^n)$ distinct assemblies to decide whether ϕ is satisfiable. Note that there are 147 distinct tiles that $\mathbb{S}_{FS}$ uses (the distinct tiles of Figure 3), and that each nondeterministic assembly assembles in time linear in the size of the input.

I will use the 8 tiles in Γ_{FS}, shown in Figure 6, to encode the input ϕ. Informally, I will encode the formula's literals in row 0, such that the literals of each clause are together and place the special clause tile to the west of each clause. I will place the tiles with ** and * west binding domains on the diagonal to the north and west of the ϕ, and I will place the tile with $|$ west binding domain as the northmost and westmost tile in the diagonal. The clear (unshaded) tiles in Figure 5(f) show the seed $S_{FS\phi}$ that encodes the 3-variable 3-clause Boolean formula $(\neg x_3 \vee \neg x_2 \vee x_0) \wedge (x_3 \vee x_2 \vee \neg x_1) \wedge (\neg x_2 \vee x_1 \vee x_0)$. The rest of Figure 5(f) shows a sample execution of $\mathbb{S}_{FS}$ on that seed. This execution assigns $x_0 = \mathit{FALSE}$ and $x_1 = \mathit{TRUE}$ in the first recursive step of the algorithm and then assigns $x_2 = \mathit{FALSE}$ and $x_3 = \mathit{TRUE}$ in the second step. In the third step, the algorithm finds that the third clause is already satisfied. Because this particular execution selected an assignment that satisfied the entire formula, the black ✓ tile is attached in the northwest corner.

Theorem 1. *Let $T_{FS} = T_{EXAM} \cup T_{SELECT} \cup T_{ROTATE} \cup T_{PREP} \cup T_{SIMPLIFY}$. Then $\mathbb{S}_{FS} = \langle T_{FS}, g_{FS}, 2\rangle$ nondeterministically decides 3-SAT with the black ✓ tile from T_{PREP} as the identifier tile. Further, for all n-variable Boolean formula ϕ, $\mathbb{S}_{FS}$ can nondeterministically form only $O^\star(1.8393^n)$ distinct assemblies.*

4 Contributions

I have presented a novel tile system $\mathbb{S}_{FS}$ that solves 3-*SAT* by nondeterministically creating $O^\star(1.8393^n)$ assemblies in parallel, for an n-variable Boolean formula. Each assembly assembles in time linear in the input size, explores some truth assignment, and attaches a special ✓ tile iff that assignment satisfies the formula. $\mathbb{S}_{FS}$ uses $147 = \Theta(1)$ distinct tile types. In some sense, $\mathbb{S}_{FS}$ implements the most complex algorithm using tiles to date. As a result, it helps bridge the gap between theoretical explorations of self-assembly, which require large tilesets to implement complex algorithms, and experimental endeavors, which have been able to combine up to 20 distinct tiles in a single experiment. Further, existing tile-inspired distributed software systems [8] can leverage $\mathbb{S}_{FS}$ directly to reduce their computational time requirements.

Acknowledgments

This work was sponsored in part by the National Science Foundation, under Grant #0937060 to the Computing Research Association for the CIFellows Project.

References

1. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight Jr., T.F., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R.: Amorphous computing. Communications of the ACM 43(5), 74–82 (2000)
2. Adleman, L.: Towards a mathematical theory of self-assembly. Tech. Rep. 00-722, Department of Computer Science, University of Southern California, Los Angeles, CA (2000)
3. Barish, R., Rothemund, P.W.K., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano Letters 5(12), 2586–2592 (2005)
4. Brun, Y.: Arithmetic computation in the tile assembly model: Addition and multiplication. Theoretical Computer Science 378(1), 17–31 (2007)
5. Brun, Y.: Nondeterministic polynomial time factoring in the tile assembly model. Theoretical Computer Science 395(1), 3–23 (2008)
6. Brun, Y.: Solving NP-complete problems in the tile assembly model. Theoretical Computer Science 395(1), 31–46 (2008)
7. Brun, Y.: Solving satisfiability in the tile assembly model with a constant-size tileset. Journal of Algorithms 63(4), 151–166 (2008)
8. Brun, Y., Medvidovic, N.: Preserving privacy in distributed computation via self-assembly. Tech. Rep. USC-CSSE-2008-819, Center for Software Engineering, University of Southern California (2008)
9. Kullmann, O.: Worst-case analysis, 3-SAT decision and lower bounds: Approaches for improved SAT algorithms. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 35, 261–313 (1997)
10. Kullmann, O.: New methods for 3-SAT decisions and worst-case analysis. Theoretical Computer Science 223, 1–72 (1999)
11. Lagoudakis, M.G., LaBean, T.H.: 2D DNA self-assembly for satisfiability. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 54, 141–154 (1999)
12. McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., Schmidt, B.: Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In: Proceedings of the AAAI Spring Symposium, Stanford, CA, USA (March 2006)
13. Monien, B., Speckenmeyer, E.: Solving satisfiability in less than 2^n steps. Discrete Applied Mathematics 10(3), 287–296 (1985)
14. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biology 2(12), e424 (2004)
15. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000), Portland, OR, USA, May 2000, pp. 459–468 (2000)
16. Schiermeyer, I.: Solving 3-satisfiability in less than 1.579^n steps. Computer Science Logic 702, 379–394 (1993)
17. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM Journal on Computing 36(6), 1544–1569 (2007)
18. Wang, H.: Proving theorems by pattern recognition. II. Bell System Technical Journal 40, 1–42 (1961)

19. Winfree, E.: Algorithmic Self-Assembly of DNA. Ph.D. thesis, California Institute of Technology, Pasadena, CA, USA (June 1998)
20. Winfree, E.: Simulations of computing by self-assembly of DNA. Tech. Rep. CS-TR:1998:22, California Institute of Technology, Pasadena, CA, USA (1998)
21. Woeginger, G.J.: Exact algorithms for NP-hard problems: a survey. Combinatorial Optimization - Eureka, You Shrink! pp. 185–207 (2003)

Optimizing Tile Concentrations to Minimize Errors and Time for DNA Tile Self-assembly Systems

Ho-Lin Chen and Ming-Yang Kao

[1] Center for Mathematics of Information,
California Institute of Technology, Pasadena, CA 91101, USA
holinc@gmail.com
[2] Department of Electrical Engineering and Computer Science,
Northwestern University, Evanston, IL 60208, USA
kao@northwestern.edu

Abstract. DNA tile self-assembly has emerged as a rich and promising primitive for nano-technology. This paper studies the problems of minimizing assembly time and error rate by changing the tile concentrations because changing the tile concentrations is easy to implement in actual lab experiments. We prove that setting the concentration of tile T_i proportional to the square root of N_i where N_i is the number of times T_i appears outside the seed structure in the final assembled shape minimizes the rate of growth errors for rectilinear tile systems. We also show that the same concentrations minimize the expected assembly time for a feasible class of tile systems. Moreover, for general tile systems, given tile concentrations, we can approximate the expected assembly time with high accuracy and probability by running only a polynomial number of simulations in the size of the target shape.

1 Introduction

Considerable modern research in science and engineering has aimed to control smaller and smaller systems in many fields, including computer science and material science. As the size of a system approaches the molecular scale, precise direct external control becomes prohibitively costly, if not impossible. As a result, bottom-up self-assembly has emerged as a rich and promising primitive for nano-technology. In particular, DNA has received much attention as a substrate for molecular self-assembly because its combinatorial nature enables the programming of molecular behaviors by choosing appropriate DNA sequences to encode information. In addition, lab techniques for the manipulation of DNA are already well developed. For these considerations, DNA self-assembly has been proposed for a variety of applications, e.g., as a means to perform computation [3,22,29], construct molecular patterns [9,12,18,19,24,33], and build nano-scale machines [5,10,13,23,25,32].

DNA tiles which self-assemble according to simple rules have been developed in lab [31] and mathematically analyzed based on the abstract tile assembly model (aTAM) proposed by Rothemund and Winfree [17]. Under this model, there is a set of square tiles with a *glue* on each of the four edges. Each glue has a certain affinity for itself called *strength.* The self-assembly process starts from a distinguished *seed structure.* Assembly proceeds as tiles attach to the partially assembled structure (initially, just the seed structure) one by one when the combined strength of matched glues between a tile and the partial structure is at least the *temperature* of the tile system. Many interesting tile systems have been designed under aTAM, including systems that build counters [1,8] and squares [11,15,17], perform Turing-universal computation [29], and produce arbitrary computable shapes [16,27]. Unfortunately, in laboratory settings, several events that aTAM does not model have been frequently observed. These events are referred to as *errors* in the tile self-assembly process. A more realistic stochastic model called the kinetic tile assembly model (kTAM) was proposed by Winfree [29] to describe the rates of these errors. The kTAM model calculates the rates for various types of attachments and detachments of tiles based on thermodynamics.

In order to make DNA tile self-assembly practical, there are two important factors that need to be minimized, namely, the error rate and the time of the assembly process. One approach to reducing the error rate of a tile assembly system [6,7,14,26,30] is to convert an existing error-prone tile system to a more robust tile system that assembles into the same shape or pattern up to scaling. These error correcting techniques increase the number of tile types by a multiplicative factor and thus are hard to implement in practice. In contrast, it is easy to change the concentrations of tiles. Therefore, it is natural to consider reducing the error rate by changing the concentrations of tiles. This approach has been studied using computer simulations and lab experiments. However, no closed-form formulas or efficient algorithms for finding the optimal tile concentrations have been previously found. It is also natural to consider changing the tile concentrations in order to minimize the assembly time. Adleman et al. [2] designed an algorithm to find tile concentrations that approximate the minimum expected assembly time within an $O(\log n)$ factor. Cheng et al. [8] showed that for partial order systems, if all tiles have equal concentrations, then the expected assembly time is proportional to the longest length of a path in the assembly order of the target shape. Also, some studies employed computer simulations to characterize the trade-offs between the time and the error rate of an assembly process [6,30].

Our Results. On the problem of minimizing the error rate, we formulate the rate of growth errors in terms of tile concentrations based on the kinetic tile assembly model. Using our formulation, we show that setting the concentration of each tile T_i proportional to the square root of the number of times T_i appears outside the seed structure in the target shape minimizes the rate of growth errors. This result holds for all rectilinear tile systems (i.e., tile systems that have the same growth directions for all tiles fixed throughout the assembly process) as well as many other systems that have been implemented in lab [4,21]. We also have simulation results showing that facet errors can significantly affect the accuracy of the optimal tile concentrations predicted by our mathematical analysis. On the problem of minimizing the assembly time, we prove that the above concentrations for minimizing the rate of growth errors also minimizes the expected assembly time for tile systems for which there is only one location for correct growth at any given time throughout the assembly process. Moreover, for general tile systems, given tile concentrations, we show that the average assembly time over a polynomial number of simulations in the size of the target shape can approximate the expected assembly time with high accuracy and probability.

The remainder of this paper is organized as follows. Section 2 describes the two tile assembly models that we use. Section 3 contains the theoretical results on minimizing the rate of growth errors. Section 4 contains the simulation results on growth errors and some discussion on facet errors. Section 5 contains the theoretical results on estimating and minimizing the expected assembly time. Section 6 concludes the paper with some open problems.

2 Two Tile Assembly Models

The Abstract Tile Assembly Model. The abstract tile assembly model was proposed by Rothemund and Winfree [17]. It extends the theoretical model of tiling by Wang [28] to include a mechanism for growth based on the physics of molecular self-assembly. Informally, a tile self-assembly system has a set of tiles, each of which is a square with glues of various types on each of the four edges. Two tiles will stick to each other if they have compatible glues. Below we present a succinct definition of this model with minor modifications for ease of explanation.

A *tile* is an oriented unit square with the *north*, *east*, *south* and *west* edges labeled from some alphabet Σ of *glues*. For each tile t, the glues of its four edges are denoted as $\sigma_N(t)$, $\sigma_E(t)$, $\sigma_S(t)$, and $\sigma_W(t)$. We describe a tile t as the quadruple $(\sigma_N(t), \sigma_E(t), \sigma_S(t), \sigma_W(t))$. Consider the triple $<T, g, \tau>$ where T is a finite set of tiles, $\tau \in \mathbf{Z}_{>0}$ is the *temperature*, and g is the *glue strength* function from $\Sigma \times \Sigma$ to $\mathbf{Z}_{\geq \mathbf{0}}$. It is assumed that for all $x, y \in \Sigma$, the inequality $x \neq y$ implies $g(x, y) = 0$ and there is a glue $null \in \Sigma$, such that $g(x, null) = g(null, x) = 0$ for all $x \in \Sigma$. A *configuration* is a map from $\mathbf{Z}^2$ to $T \bigcup \{empty\}$, where *empty* is a special symbol indicating the absence of any tile.

A *tile system* is a quadruple $\mathbf{T} = <T, s, g, \tau>$, where T, g, τ are as above and s is a special configuration called the *seed structure*. Let C and D be two configurations. Suppose that there exist some $t \in T$ and some $(x, y) \in \mathbf{Z}^2$ such that $D = C$ except that at (x, y), $C(x, y) = null$ and $D(x, y) = t$. Let $f_{N,C,t}(x, y) = g(\sigma_N(t), \sigma_S(C(x, y + 1))$. Informally, $f_{N,C,t}(x, y)$ is the strength of the bond on the north edge of t in configuration C. We define $f_{S,C,t}(x, y), f_{E,C,t}(x, y)$ and $f_{W,C,t}(x, y)$ similarly. Then tile t is *attachable* to C at position (x, y) iff $f_{C,t}(x, y) \equiv f_{N,C,t}(x, y) + f_{S,C,t}(x, y) + f_{E,C,t}(x, y) + f_{W,C,t}(x, y) \geq \tau$. We write $C \rightarrow_{\mathbf{T}} D$ to denote the transition from C to D by attaching a tile to C at position (x, y). Informally, $C \rightarrow_{\mathbf{T}} D$ iff D can be obtained from C by adding a tile t such that the total strength of interaction between t and C is at least τ. A *terminal assembly* is a configuration A such that there is no configuration B for which $A \rightarrow_{\mathbf{T}} B$.

When a tile t attaches to configuration C at position (x, y), the edges U of t with $f_{U,C,t}(x, y) > 0$ are called the *input* edges; all other edges are called the *output* edges. A tile system is *rectilinear* if there is a unique terminal assembly that can be reached starting from the seed structure, each tile t has the same input and output edges every time it attaches, and all tiles have the same input and output edges.

The Kinetic Tile Assembly Model. According to the abstract tile assembly model, a tile t attaches at a position (x, y) in a configuration C iff the total strength $f_{C,t}(x, y)$ of the matched glues between C and t is at least τ, and any tiles that attached never fall off. In practice, tiles may attach with a weaker binding strength, and tiles that already attached may fall off. These events can cause the tile system to behave differently from the abstract tile assembly model. We treat these deviations as errors and try to minimize the probability of these events. In this paper, we use the kinetic tile assembly model proposed by Winfree [29] to model the *forward* and *reverse* rates, which are the rates at which a tile attaches to and falls off from a specific position, respectively. This model computes these rates as functions of thermodynamic parameters as follows:

1. The concentrations of the tiles are held constant throughout the self-assembly process.
2. The only two reactions allowed are single tiles attaching to and dissociating from a configuration.
3. The forward rate for tile T_i is $k_f c_i$, where k_f is a constant and c_i is the concentration of tile T_i. This notation is used throughout this paper.
4. The reverse rate for a tile t attached to configuration C at position (x, y) to fall off is $k_f e^{-bG_{se}}$, where k_f and G_{se} are constants and $b = f_{C,t}(x, y)$ is the total strength of the matched glues between t and C.

Here, the parameters k_f and k_f give the time scale of the self-assembly. The value of G_{se} is determined by the binding strength of the sticky ends of DNA tiles. We use $c_{\max}$ and $c_{\min}$ to denote the maximum and minimum concentrations allowed in the tile system. If one wants the tile system to assemble according to the abstract tile assembly model most of the time, then the following two conditions need to hold. First, if the total binding strength between a tile and the original configuration it just attached to is less than τ, then the tile must fall off quickly, i.e.,

$$k_{\mathrm{f}} e^{-(\tau-1)G_{\mathrm{se}}} \gg k_f c_{\max}.$$

Second, if a tile t is attachable to a position in C, then the forward rate at which it attaches should be greater than the reverse rate at which it falls off, i.e.,

$$k_f c_{\min} > k_f e^{-\tau G_{se}}.$$

In practice, since each tile may have a slightly different value for the parameter k_f and the strength of each glue may vary, one often needs to set k_f and G_{se} (by changing an experiment's temperature) such that

$$k_f c_{\min} \gg k_f e^{-\tau G_{se}}.$$

For the remainder of the paper, we assume

$$k_f e^{-(\tau-1)G_{se}} \gg k_f c_{\max} > k_f c_{\min} \gg k_f e^{-\tau G_{se}}.$$

We also assume that our seed structure is made by some other processes (e.g., DNA origami [18]) and its tiles never fall off.

3 Minimizing the Error Rate

In this section, we consider the problem of changing the concentrations of tiles to minimize the failure probability (i.e., error rate) for a rectilinear tile system. There are three types of errors in tile self-assembly. A *growth error* refers to an incorrect tile attaching at a position instead of the correct tile [30]. A *facet error* refers to an incorrect tile attaching at a position where no tile is supposed to attach [6]. A *nucleation error* refers to single tiles attaching to each other to form a lattice without the seed structure [20]. In this section, we only consider minimizing growth errors.

We want to compute the probability that a tile T_j causes a growth error by attaching at a position (x, y) where only T_i can attach with total binding strength at least τ. First, T_i can attach at that position at rate $k_f c_i$. Once T_i has attached, the probably of it falling off is negligible since $k_f c_{\min} \gg k_f e^{-\tau G_{se}}$. Second, T_j can attach at that position at rate $k_f c_j$. Once T_j has attached, it can fall off at rate $k_f e^{-(\tau-m)G_{se}}$, where m is the total strength of the mismatched glues between T_i and T_j on their input edges. T_j can also get locked in place and cause an error due to the attachment of one or more adjacent tiles. The rate r at which T_j gets locked in place may vary with the features in the partially assembled shape near position (x, y) such as long facets. In this paper, we assume that r is the same for all positions (x, y) and tiles t. The allowable reactions related to T_i and T_j are summarized in Figure 1. From the above description of reaction rates, we know that at a given position (x, y) where tile T_i is supposed to attach, the probability of having a growth error caused by T_j is $\frac{c_j}{c_i}\epsilon_{ij}$. Here, $\epsilon_{ij} = \frac{r}{r+k_f e^{-(\tau-m)G_{se}}}$, where m is the total number of mismatches between the input sides of tiles T_i and T_j. The value of ϵ_{ij} is roughly at the order of $e^{-mG_{se}}$ since $r \leq 2k_f c_{\max}$. Therefore, at position (x, y), the total probability of a growth error is

$$\frac{\sum_{j \neq i} \epsilon_{ij} c_j}{c_i}.$$

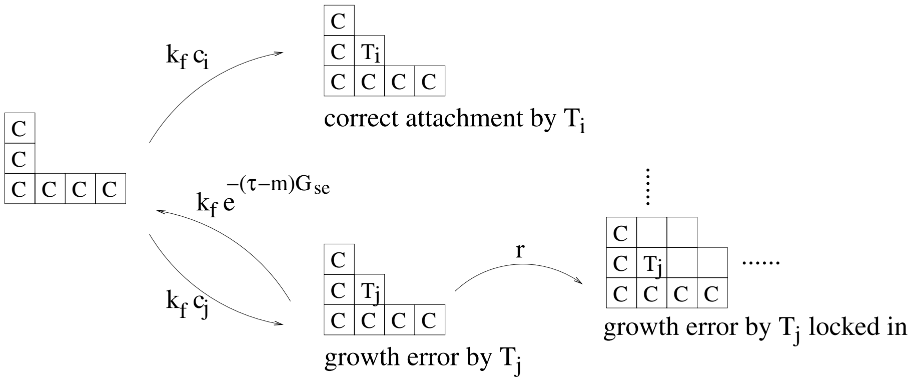


Fig. 1. A Markov chain describing attachments of T_i and T_j, where C indicates a correct tile

For a self-assembly process, the error rates at different positions depend on each other. However, if one wants to have a high probability of success, one almost always needs to set the experimental condition such that the error rate at each position is much smaller than $1/n$, where n is the total number of tiles in the desired terminal assembly. In this case, minimizing the sum of error rates over all positions is a good approximation of minimizing the actual overall error rate of the tile assembly system. Thus, in the remainder of this section, we will minimize

$$\sum_i N_i\left(\frac{\sum_{j\neq i}\epsilon_{ij}c_j}{c_i}\right), \qquad (1)$$

where N_i is the number of positions outside the seed structure to which T_i is supposed to attach.

Theorem 1. *For a rectilinear tile system with a unique terminal assembly, the error rate (i.e., probability of failure) is minimized when the concentration of each tile T_i is proportional to $\sqrt{N_i}$, where N_i is the number of times tile T_i appears outside the seed structure in the correct terminal assembly of the tile system.*

Proof. From Equation 1, we can scale the tile concentrations c_i without loss of generality such that $\sum_i c_i = 1$, and we need to solve the following minimization problem:

$$\text{Minimize} \sum_i N_i\left(\frac{\sum_{j\neq i}\epsilon_{ij}c_j}{c_i}\right),$$
$$\text{subject to} \sum_i c_i \;=\; 1.$$

The Lagrange multiplier for this minimization problem is

$$\Lambda \;=\; \sum_i N_i\left(\frac{\sum_{j\neq i}\epsilon_{ij}c_j}{c_i}\right) \;+\; \lambda\left(\sum_i c_i - 1\right).$$

We need to solve

$$\frac{\partial\Lambda}{\partial c_i} \;=\; -N_i\left(\frac{\sum_{j\neq i}\epsilon_{ij}c_j}{c_i^2}\right) \;+\; \sum_{j\neq i} N_j\frac{\epsilon_{ij}}{c_j} \;+\; \lambda \;=\; 0 \quad \text{for all } i \qquad (2)$$

and

$$\sum_i c_i \;=\; 1.$$

Simplifying Equation 2, we obtain

$$\sum_{j\neq i}\left(-N_i\frac{\epsilon_{ij}c_j}{c_i^2} + N_j\frac{\epsilon_{ij}}{c_j}\right) \;+\; \lambda \;=\; 0 \quad \text{for all } i,$$

and consequently the error rate is minimized when

$$c_i = \frac{\sqrt{N_i}}{\sum_j \sqrt{N_j}}.$$

Two points about the proof of Theorem 1 are worth noticing. First, the error rate only depends on the ratio between the tile concentrations. Specifically, the error rate is minimized when the concentration of T_i is proportional to $\sqrt{N_i}$ even when we vary each c_i between $c_{\max}$ and $c_{\min}$. Second, the same proof can apply to all tile systems that satisfy $\epsilon_{ij} = \epsilon_{ji}$ for all i, j. Hence Theorem 1 is valid for other systems already implemented in lab such as zig-zag ribbons [21] and counters seeded by origami [4].

4 Simulation Results for Theorem 1

We used a software called xgrow developed in Erik Winfree's lab to simulate four tile systems to determine their error rates under different tile concentrations. To obtain a good estimate of the error rate of a tile system, we would choose our parameters such that errors can be frequently observed. However, in most tile systems, if we use such parameters, we will reach some configuration very different from the terminal assembly predicted by the abstract tile assembly model. Since our prediction of the optimal tile concentrations depends on the terminal assembly, we made a design decision to perform simulations on tile systems for which each error only affects one position of the terminal assembly.

We simulated four tile systems named as A_1, A_2, B_1, and B_2. Each of the four systems operates at $\tau = 2$ and only has two tiles X and Y shown in Figure 2(a) beside the seed structure. The only difference between the four systems is their seed structures. The seed structures of tile systems A_1 and A_2 are shown in Figure 2(b). The lengths of their seed structures are adjusted such that $N_X : N_Y = 25 : 1$ and $64 : 1$ for A_1 and A_2, respectively, where N_X and N_Y are the numbers of positions X and Y appear in the terminal assembly. The seed structures of tile systems B_1 and B_2 are shown in Figure 2(c). The lengths of their seed structures are also adjusted such that $N_X : N_Y = 25 : 1$ and $64 : 1$ for B_1 and B_2, respectively. These systems are rectilinear with all tiles having their input edges on the south and east edges. Since tiles X and Y have the same output edges, when an error happens, the error only affects the position where the erroneous tile is attached. The unique terminal assemblies and example configurations generated by simulations of tile systems A_1 and B_1 are shown in Figures 3 and 4, respectively. Theorem 1 predicts that the rate of growth errors is minimized at $c_X : c_Y = 5 : 1$ for systems A_1 and B_1, and at $c_X : c_Y = 8 : 1$ for systems A_2 and B_2.

The simulation results are shown in Figure 5. In systems A_1 and A_2, the optimal tile concentration ratios are $2.5 : 1$ and $3 : 1$, respectively. In systems B_1 and B_2, the optimal tile concentration ratios are roughly $7.5 : 1$ and $15 : 1$, respectively. The major reason causing these simulation results to deviate from the predictions made by Theorem 1 appears to be the facet errors. Since tiles X and Y both have glue 0 on the south edge, having a long horizontal facet may introduce a large number of facet errors. For systems A_1 and A_2, notice that long horizontal facets are generated because tile Y (colored yellow) has lower concentrations and grows slower than X. An example configuration for system A_1 that demonstrates these horizontal facets is shown in Figure 3. Such undesirable facets become longer and more when we increase the ratio between c_X and c_Y. Therefore, the actual optimal tile concentrations are biased towards having more Y than predicted by Theorem 1. For systems B_1 and B_2, each terminal assembly is separated into a left portion and a right portion by the seed structure, as shown in Figure 4. Horizontal facets can only be generated in the left portion, where all tiles should be X. Hence, we can reduce facet errors by decreasing the concentration of tile Y, and thus the actual optimal tile concentrations are biased towards having fewer Y than predicted by Theorem 1.

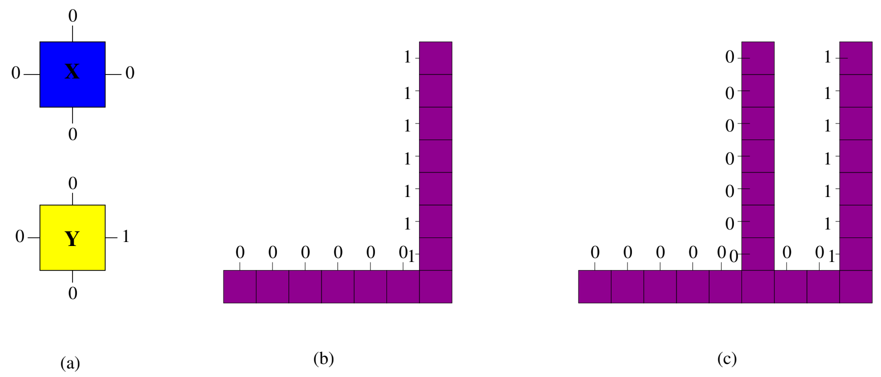


Fig. 2. (a) Tiles X and Y, where all glues have strength 1. (b) An L-shaped seed structure for systems A_1 and A_2. (c) A seed structure with two vertical facets and one horizontal facet for systems B_1 and B_2.

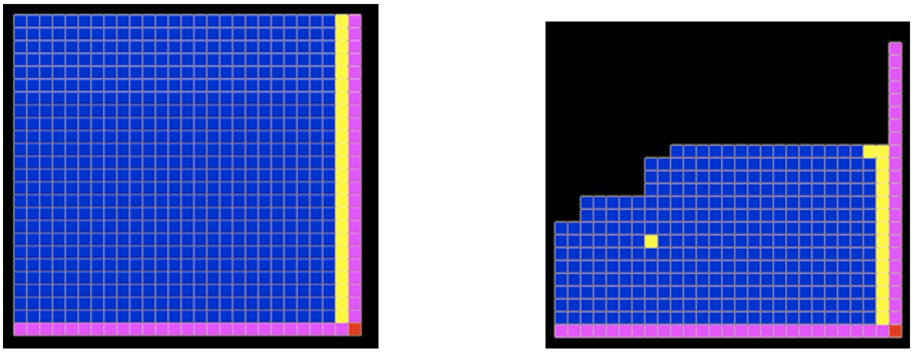


Fig. 3. The left figure is the unique target terminal assembly for system A_1. The right figure is an example configuration for system A_1 generated by an xgrow simulation.

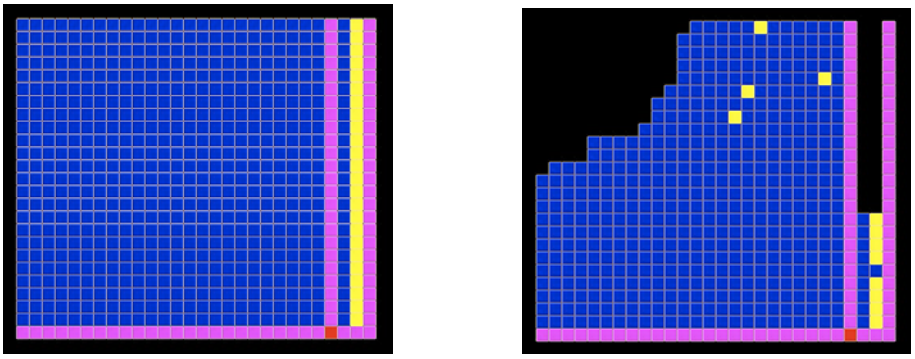
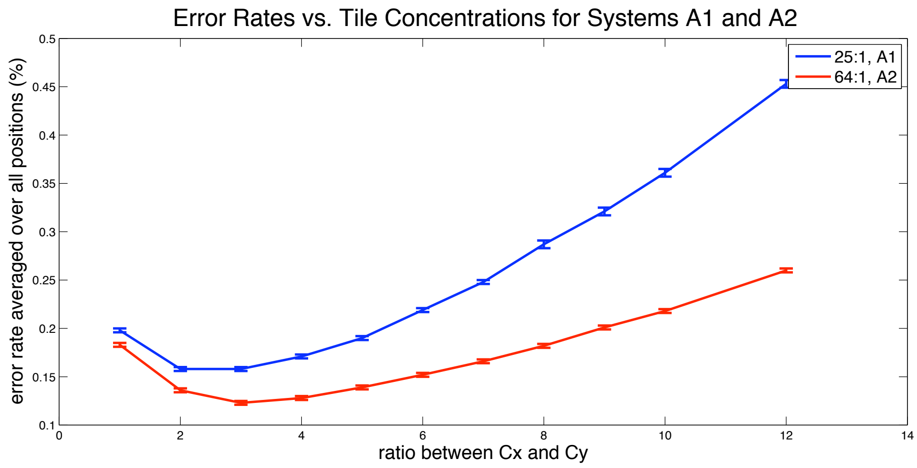


Fig. 4. The left figure is the unique target terminal assembly for system B_1. The right figure is an example configuration for system B_1 generated by an xgrow simulation.



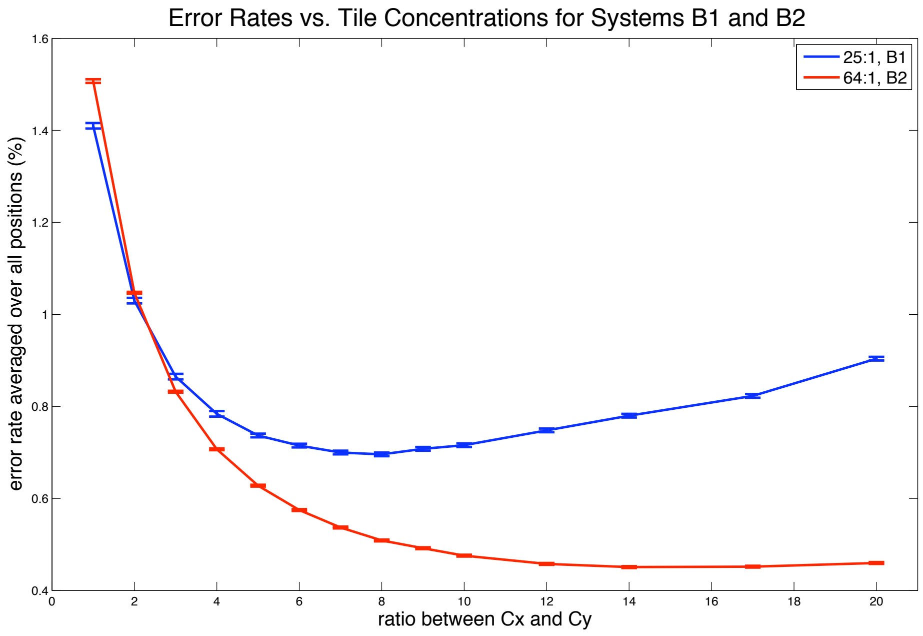


Fig. 5. Plots of error rates vs. the ratios between tile concentrations. Each data point represents $m = 20,000$ simulations. The simulations use $G_{se} = 9$ for systems A_1 and A_2, $G_{se} = 11$ for systems B_1 and B_2, $c_X + c_Y = e^{-16}$. Error bars show two standard deviations of the errors, computed using $\sigma = \sigma_{\text{simulation}}/\sqrt{m}$.

5 Minimizing the Expected Assembly Time

This section assumes that only the correct tiles can attach and any tile that has already attached never falls off. We minimize the expected assembly time by varying the tile concentrations.

Theorem 2. *Consider any tile system with the four properties that*

1. *at any given time, only one location can have a tile attach correctly,*
2. *only the correct tiles can attach,*
3. *any tile that has already attached never falls off, and*
4. *there is a unique terminal assembly.*

Assume that the total tile concentration is $\sum_i c_i = 1$*. Setting* $c_i = \frac{\sqrt{N_i}}{\sum_j \sqrt{N_j}}$ *minimizes the expected assembly time of the tile system.*

Proof. Omitted due to space constraints.

In settings that are more general than Theorem 2 assumes, the optimal tile concentrations to minimize the expected assembly time may significantly deviate from the c_i's determined in Theorem 1.

For general tile assembly systems, we do not know how to analytically find the optimal tile concentrations to minimize the expected assembly time. However, we show in Theorem 3 below that given a set of tile concentrations, only a polynomial number of simulations is required in order to approximate the expected assembly time with high accuracy and probability.

Lemma 1. *Consider any tile system for which Properties 2 through 4 in Theorem 2 hold but there is no assumption on whether tiles can attach only one by one or in parallel. If the assembly process of the tile system takes expected time* S*, then for any* $\epsilon > 0$*, the average of the assembly times over* $48S^2\frac{1}{\epsilon}$ *simulations of the assembly process will be between* $S - \epsilon$ *and* $S + \epsilon$ *with probability at least* $3/4$*.*

Proof. Omitted due to space constraints.

Theorem 3. *Consider any tile system for which Properties 2 through 4 in Theorem 2 hold but there is no assumption on whether tiles can attach only one by one or in parallel. Let* n *be the number of positions outside the seed structure in the terminal assembly. If the assembly process of the tile system takes expected time* S*, then for any* $\epsilon > 0$*, the average of the assembly times over* $O(n^4\frac{1}{\epsilon\ c_{\min}^2})$ *simulations of the assembly process will be between* $S-\epsilon$ *and* $S+\epsilon$ *with probability at least* $3/4$*.*

Proof. Omitted due to space constraints.

6 Further Research

In Section 3, we gave closed-form formulas to minimize the growth errors by varying the concentration of each tile. In Section 4, we found in simulations that facet errors are also an important factor that needs to be considered in order to minimize the error rate in lab implementations. At the theoretical level, it is open to find closed-form formulas or efficient algorithms to minimize the facet errors by varying the tile concentrations.

In Section 5, we gave closed-form formulas to minimize the expected assembly time for a feasible class of tile systems by varying the tile concentrations. For general tile systems, the best known algorithm can compute an $O(\log n)$-approximation of the minimum expected assembly time [2]. It is of interest to determine whether one can compute the precise minimum expected assembly time or an estimate with a better approximation factor than $O(\log n)$. Given tile concentrations, we showed that simulations can accurately predict the expected assembly time with high probability. The computing time it takes to run the required simulations is polynomial in the size of the terminal assembly, not the tile system itself. Since the size of a tile system is normally smaller than that of its terminal assembly, it would be useful if one can approximate the expected assembly time just by analyzing the tile system and some succinct features of the terminal assembly (e.g., the number of times each tile appears outside the seed structure in the terminal assembly) in time polynomial in the size of the tile system.

References

1. Adleman, L., Cheng, Q., Goel, A., Huang, M.-D.: Running time and program size for self-assembled squares. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, pp. 740–748 (2001)
2. Adleman, L., Cheng, Q., Goel, A., Huang, M.-D., Kempe, D., Moisset de Espans, P., Rothemund, P.: Combinatorial optimization problems in self-assembly. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 23–32 (2002)
3. Barish, R.D., Rothemund, P.W.K., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano Letters 5(12), 2586–2592 (2005)
4. Barish, R.D., Schulman, R., Rothemund, P.W.K., Winfree, E.: An information-bearing seed for nucleating algorithmic self-assembly. Proceedings of the National Academy of Sciences 106, 6054–6059 (2009)
5. Bishop, J., Klavins, E.: An improved autonomous DNA nanomotor. Nano Letters 7(9), 2574–2577 (2007)
6. Chen, H.-L., Goel, A.: Error free self-assembly using error prone tiles. In: Proceedings of the 10th International Meeting on DNA Based Computers, pp. 62–75 (2004)
7. Chen, H.-L., Luhrs, C., Goel, A.: Dimension augmentation and combinatorial criteria for efficient error-resistant DNA self-assembly. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 409–418 (2008)
8. Cheng, Q., Goel, A., Moisset, P.: Optimal self-assembly of counters at temperature two. In: Proceedings of the 1st Conference on Foundations of Nanoscience: Self-Assembled Architectures and Devices, pp. 62–75 (2004)
9. Dietz, H., Douglas, S., Shih, W.: Folding DNA into twisted and curved nanoscale shapes. Science 325, 725–730 (2009)
10. Ding, B., Seeman, N.: Operation of a DNA robot arm inserted into a 2D DNA crystalline substrate. Science 384, 1583–1585 (2006)
11. Doty, D.: Randomized self-assembly for exact shapes. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 85–94 (2009)
12. Douglas, S., Dietz, H., Liedl, T., Hogberg, B., Graf, F., Shih, W.: Self-assembly of DNA into nanoscale three-dimensional shapes. Nature (459), 414–418 (2009)

13. Green, S., Bath, J., Turberfield, A.: Coordinated chemomechanical cycles: a mechanism for autonomous molecular motion. Physical Review Letters (101), 238101 (2008)
14. Sahu, S., Reif, J., Yin, P.: Compact error-resilient computational DNA tiling assemblies. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 293–307. Springer, Heidelberg (2005)
15. Kao, M.-Y., Schweller, R.: Reducing tile complexity for self-assembly through temperature programming. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 571–580 (2006)
16. Lagoudakis, M., LaBean, T.: 2D DNA self-assembly for satisfiability. In: Proceedings of the 5th DIMACS Workshop on DNA Based Computers. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 54, pp. 141–154 (1999)
17. Rothemund, P., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, pp. 459–468 (2000)
18. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature (440), 297–302 (March 2006)
19. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLOS Biology 2, 424–436 (2004)
20. Schulman, R., Winfree, E.: Programmable control of nucleation for algorithmic self-assembly. In: Proceedings of the 10th International Meeting on DNA Based Computers, pp. 319–328 (2004)
21. Schulman, R., Winfree, E.: Self-replication and evolution of DNA crystals. In: Proceedings of the 5th European Conference on Artificial Life, pp. 734–743 (2005)
22. Seelig, G., Soloveichik, D., Zhang, D., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314, 1585–1588 (2006)
23. Sherman, W.B., Seeman, N.C.: A precisely controlled DNA bipedal walking device. Nano Letters 4, 1203–1207 (2004)
24. Shih, W.M., Quispe, J.D., Joyce, G.F.A.: A 1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron. Nature (427), 618–621 (2004)
25. Shin, J.-S., Pierce, N.A.: A synthetic DNA walker for molecular transport. Journal of American Chemistry Society 126, 10834–10835 (2004)
26. Soloveichik, D., Cook, M., Winfree, E.: Combining self-healing and proofreading in self-assembly. Natural Computing (7), 203–218 (2008)
27. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM Journal on Computing 36, 1544–1569 (2007)
28. Wang, H.: Proving theorems by pattern recognition ii. Bell Systems Technical Journal 40, 1–42 (1961)
29. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology, Pasadena (1998)
30. Winfree, E., Bekbolatov, R.: Proofreading tile sets: Error correction for algorithmic self-assembly. In: Proceedings of the 9th International Meeting on DNA Based Computers, pp. 126–144 (2003)
31. Winfree, E., Liu, F., Wenzler, L., Seeman, N.: Design and self-assembly of two-dimensional DNA crystals, 6 pages. Nature (394), 539–544 (August 1998)
32. Yurke, B., Turberfield, A., Mills Jr., A., Simmel, F., Neumann, J.: A DNA-fuelled molecular machine made of DNA. Nature (406), 605–608 (August 2000)
33. Zhang, Y., Seeman, N.: Construction of a DNA-truncated octahedron. Journal of American Chemical Society 116(5), 1661 (1994)

Scalable, Time-Responsive, Digital, Energy-Efficient Molecular Circuits Using DNA Strand Displacement*

Ehsan Chiniforooshan, David Doty, Lila Kari, and Shinnosuke Seki

Univ. of Western Ontario, Dept. of Computer Science, London, Canada
{ehsan,ddoty,lila,sseki}@csd.uwo.ca

Abstract. We propose a novel theoretical biomolecular design to implement any Boolean circuit using the mechanism of DNA strand displacement. The design is *scalable*: all species of DNA strands can in principle be mixed and prepared in a single test tube, rather than requiring separate purification of each species, which is a barrier to large-scale synthesis. The design is *time-responsive*: the concentration of output species changes in response to the concentration of input species, so that time-varying inputs may be continuously processed. The design is *digital*: Boolean values of wires in the circuit are represented as high or low concentrations of certain species, and we show how to construct a single-input, single-output signal restoration gate that amplifies the difference between high and low, which can be distributed to each wire in the circuit to overcome signal degradation. This means we can achieve a digital abstraction of the analog values of concentrations. Finally, the design is *energy-efficient*: if input species are specified ideally (meaning absolutely 0 concentration of unwanted species), then output species converge to their ideal concentrations at steady-state, and the system at steady-state is in (dynamic) equilibrium, meaning that no energy is consumed by irreversible reactions until the input again changes.

1 Introduction

Biomolecular circuits, due to natural compatibility with the materials of life, could change the way diseases are diagnosed or medicine is delivered, through bottom-up, site-specific biochemical information processing. While there is no shortage of theoretical proposals and experimental implementations [1–7, 9, 11], the state of the art in biomolecular circuits remains far behind its electronic equivalent.

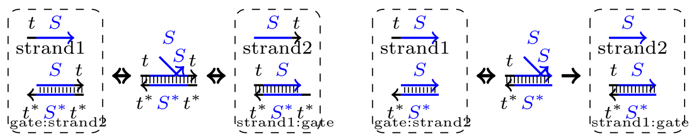
In this paper, we propose a new theoretical design, focusing on one particular molecular primitive as our sole "basic operation" from which to compose complex circuits: *toehold-mediated DNA branch migration and strand displacement*,

* This research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant R2824A01 and the Canada Research Chair Award in Biocomputing to Lila Kari, and by the NSF Computing Innovation Fellowship grant to David Doty.

or simply *strand displacement*. The idea of the basic strand displacement reaction is shown in Figure 1. DNA strand displacement as a tool for nanoengineering was introduced by Yurke, Turberfield, Mills, Simmel, and Neumann [10], and its use as a primitive for building circuits was pioneered by Seelig, Soloveichik, Zhang, and Winfree [7], and subsequently improved and simplified by Zhang, Turberfield, Yurke, and Winfree [11] and Qian and Winfree [6]. Compared to some designs using other "molecular primitives" such as restriction enzymes, strand displacement has the advantage that it requires the design of no new molecules other than single-stranded DNA complexes, which are easily and cheaply available through mail-order.

Our construction achieves four properties desirable of robust circuit designs. We do not claim these properties to represent the sole criteria by which to judge molecular circuit designs, but we believe they are evidently advantageous. We emphasize that achieving any of these properties in isolation would not be a novel contribution, but to our knowledge this is the first DNA circuit design to incorporate all four simultaneously. The properties are as follows.

scalable: This is a nebulous word with many connotations. Our particular usage refers to the definition given by Qian and Winfree in [6], whose construction overcomes a specific inhibiting factor that prevents large-scale fabrication of molecular circuits: the need to prepare and purify different molecular species in separate test tubes. Since a Boolean circuit with hundreds of gates may require thousands of distinct molecular species, it is no small advantage to be able to mix all of them together in a single tube and conduct the necessary



(a) Reversible (b) Irreversible

Fig. 1. Example of basic DNA strand displacement reactions. Figure 1a is the reversible reaction strand1 + gate:strand2 $\rightleftharpoons$ strand1:gate + strand2, and Figure 1b is the irreversible reaction strand1 + gate:strand2 $\rightarrow$ strand1:gate + strand2. Each strand is represented as an arrow indicating 5'-to-3' orientation. t and S_1 are finite strings over $\{A, C, G, T\}$, and t^* and S^* are their 3'-to-5'-oriented Watson-Crick complements. The "toehold" region t is short enough (about 5 nucleotides) that it cannot provide sufficient binding strength to allow two strands to hybridize stably. Only if the longer (say, at least 15 nucleotides) "recognition" region S matches between strand1 and the base strand gate can strand1 displace strand2. Displacement occurs via an unbiased random walk but proceeds quickly compared to the time taken for strands to find each other in solution, so a successful displacement is modeled as occurring instantaneously upon binding of the toehold.

preparation steps solely on that one tube; this is what we mean by "scalable".[1] The particular mechanism we utilize to achieve scalability is the same used by Qian and Winfree. Briefly, it involves creating double-stranded complexes from single-stranded hairpin precursors that are cleaved with (naturally occurring) restriction enzymes, possibly leaving short sticky ends to serve as toeholds for future strand displacement.

time-responsive: This property is achieved by Goel and Ibrahimi [1] utilizing restriction enzyme technology (apparently requiring new restriction enzymes to be designed). Qian and Winfree [6] describe time-responsiveness as an open problem for their particular motif, known as seesaw gates, which are built from strand displacement cascades. Informally, a circuit is time-responsive if, supposing that the inputs to the circuit change after the initial computation, then the output is re-computed to reflect the new inputs. Time-responsiveness of individual gates is key to constructing recurrent circuits that use feedback loops to implement memory storage devices, such as latches and flip-flops. Even for feed-forward circuits, time-responsiveness is an intuitively appealing property. For instance, a circuit could constantly monitor the state of a cell and release drugs in response to a temporary malady, then inhibit the release as the malady disappears.

energy-efficient: This property is a definition of our own device, but it or something approximating it seems essential in robust circuit implementations. Our definition is that inputs given ideally result in eventual migration to a steady state in which 1) outputs are also ideal, and 2) this steady state is in equilibrium, maintained without any expenditure of energy to power irreversible reactions. This is a dynamic equilibrium, as some reversible reactions are always taking place. In our system (as in many others), Boolean values of all wires in the circuit, including input, output, and intermediate gate-connecting wires, are represented as high or low concentrations of certain chemical species. As each wire w in the abstract circuit being simulated can take on the values 0 and 1, this wire is associated to two chemical species $\mathtt{0}_w$ and $\mathtt{1}_w$. Ideally, to represent the bit $b \in \{0,1\}$, b_w is present and $\overline{b}_w$ is absent (where $\overline{b} = 1 - b$), the so-called *dual-rail* convention. Our design ensures energy efficiency because energy is expended only to change non-ideal wires (those with positive concentration of b_w when $\overline{b}$ is the correct bit for that wire given the inputs) to ideal, but no energy is expended to maintain a wire's correctness once it reaches an ideal state. We note that this definition assumes perfectly irreversible reactions are possible. In reality, more energy is required to drive a reaction the greater its ratio of forward to reverse rates; hence a completely irreversible reaction requires infinite energy. A better quantitative definition of energy efficiency would take into account this partial reversibility of all chemical reactions, including our "irreversible" reactions, when measuring energy use.

[1] Although we have not formally defined the notion, it seems reasonable, for instance, to define a "scalable" molecular circuit to be one whose preparation involves at most a constant number of preparation steps (other than the original design of the molecules themselves), regardless of the size of the circuit.

digital: By this we mean that the circuit employs signal restoration to obtain a digital abstraction of what are fundamentally analog concentration values. Since Boolean values are represented by high or low concentrations of certain chemical species, to correct for non-ideal inputs, as well as the natural signal degradation suffered by the logic gates, it is desirable to move high concentrations higher and low concentrations lower before feeding the values as input to the next gate in the circuit. We achieve this by designing a single-input, single-output restoration gate (to be "spliced" into every wire in the logical circuit) such that, if i is the input wire and o is the output wire, then $[\mathtt{1}_o]/[\mathtt{0}_o] = ([\mathtt{1}_i]/[\mathtt{0}_i])^2$, where $[A]$ denotes the concentration of species A. For instance, if a wire's species have combined concentration 100, and if $[\mathtt{1}_i] = 60$ and $[\mathtt{0}_i] = 40$, then at steady-state, $[\mathtt{1}_o] \approx 69.23$ and $[\mathtt{0}_o] \approx 30.77$ (since $69.23/30.77 \approx 2.25 = 1.5^2 = (60/40)^2$). By serially cascading a small number of such gates together we can amplify even very weak signals, since n cascaded gates amplify a ratio of r to r^{2^n}. For instance, to transform a ratio of $0.6/0.4$ to $> 0.99999/0.00001$ requires only 5 restoration gates.

Our design has been simulated using the idealized model of DNA strand displacement kinetics described in [6, 8], and the simulation agrees with the properties claimed above. An important future project is to experimentally validate that such a construction is possible.

2 Construction

This section describes the details of our proposed design. Recall the dual-rail convention that our design employs: each wire w in the circuit will be represented by two chemical species (DNA strands) $\mathtt{0}_w$ and $\mathtt{1}_w$, and the Boolean gates we use work as long as the ratio of the concentration of the correct input bit species to the incorrect input bit species is "sufficiently high". Our design involves the construction of three types of gates, the first two Boolean and the third analog: 2-input/1-output Boolean NAND gates (output bit is 0 if and only if both input bits are 1), 1-input/2-output Boolean fan-out gates (both output bits are equal to input bit), and 1-input/1-output analog signal restoration gates (the difference in concentration between species $\mathtt{0}_w$ and $\mathtt{1}_w$ representing the input wire w is amplified for the output wire species). Every Boolean function can be computed by the composition of NAND gates, and through the appropriate composition of fan-out gates we may assume that each intermediate wire in the circuit connects exactly two gates, and that each input and output wire is connected to exactly one gate. We then place a signal restoration gate (or more in series) on each wire of the circuit as needed. This is necessary since the Boolean gates suffer some signal degradation, and since inputs may not be specified ideally. The conversion of an example circuit is shown in Figure 2. We choose NAND gates for the sake of concreteness, but it is easy to modify our design to implement any of the sixteen possible 2-input/1-output logic gates.

Our design of the three gates is described in terms of sets of abstract chemical reactions implemented by DNA strand displacement in a similar fashion

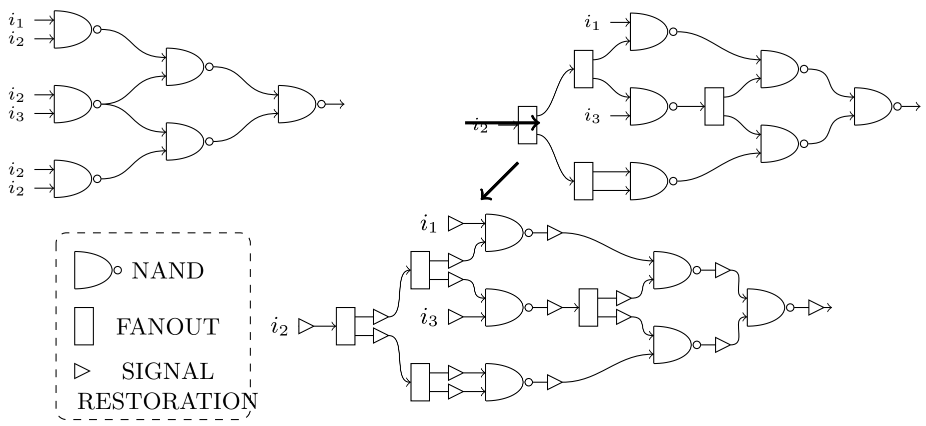


Fig. 2. A circuit with 6 Boolean NAND gates and 3 inputs, converted to add Boolean fan-out gates to allow fan-out from inputs and NAND gates, as well as analog signal restoration gates between all Boolean gates

to [8]. Each species in the reactions below is represented by single-stranded DNA molecules of the same form as strand1 in Figure 1;[2] since they all have the same format, arbitrarily large circuits may be "wired" together. Our method of implementing abstract chemical reactions with DNA strand displacement reactions is inspired by, but subtly different from, that of [8]. As in [8], a single "high-level" abstract reaction is simulated by more than one "underlying" DNA strand displacement reaction. The primary difference is the need for a direct implementation of a termolecular reaction (a reaction with three reactants) in which no irreversible "underlying implementation reaction" is allowed to occur unless all three reactants of the "high-level reaction" are present. This allows us to conclude that if no abstract high-level reactions are possible, then no irreversible, energy-consuming implementation reactions are possible. This ensures energy efficiency so long as we design the high-level reactions so that they cannot occur at steady-state.

We do not specify the rate constant of any reaction. For the low-level strand displacement reactions all rate constants are assumed to be equal, following the kinetic model of [8] in which toehold length determines the rate constant of a strand displacement reaction (all our toeholds are equal length). In implementation using DNA strands, the abstract high-level reactions will not have identical rate constants, but the implementation is robust to small differences in rate constants among the high-level reactions. For instance, it suffices if all of these rate constants are within an order of magnitude of each other, which they are given our implementation.

The time-responsiveness of the design is evident by inspection of the reactions below. Informally, it follows from the fact that each input to a gate is catalytic in

[2] Other intermediate species not shown in the section but used in the DNA strand implementation in Section 2.2 are not necessarily of this form.

the reactions associated with that gate and the fact that the total number of "bit molecules" $\mathsf{0}_w$ and $\mathsf{1}_w$ associated with a wire w is constant; reactions only change $\mathsf{0}_w$ to $\mathsf{1}_w$ and vice-versa. Fuel molecules are indeed consumed for each reaction. Some consumed fuel species are explicit in the reactions discussed below (such as diff_o or $\mathsf{0f}_o$), but some fuel is explicit only in the "underlying implementation reactions" of DNA strand displacement and does not even appear as an abstract chemical species below. For each set of reactions below the fuel efficiency of the reactions is justified in the sense that none of the reactions can occur at the steady-state concentrations assuming ideal input species. We must take care when implementing these reactions with DNA strand displacement to ensure that all underlying implementation reactions that occur at steady-state are reversible, but this is not discussed in this section. The digital abstraction is achieved by one particular signal restoration gate discussed in Section 2.1, intended to be distributed throughout the circuit in between the other gates. The scalability of our design is achieved in a similar fashion to that used by Qian and Winfree [6], although slightly more complex in making use of multiple hairpin precursor DNA strands. We omit a full explanation here, which can be found in the full version of this extended abstract.

2.1 Design of Abstract Chemical Reactions for Gates

Given the circuit structure proposed above (NAND gates, fan-out gates, and signal restoration gates), we may assume that each wire in the circuit is the input of exactly one gate and the output of exactly one gate. If each gate g in the circuit is given a unique identifier id_g, then a wire connecting gate g to g' is uniquely identified by the pair $w = (\mathsf{id}_g, \mathsf{id}_{g'})$. We therefore assume each wire has a unique identifier that indicates which two gates it connects. Ultimately, this wiring information will be encoded by DNA recognition regions, but for now we describe the gate operation in terms of wires labeled with abstract values. The concentrations of the species associated with the input wires are assumed to be under external control. We take care to ensure that all wire species are catalytic (not consumed) in the reactions of gates for which they are an *input*; this ensures time-responsiveness of each of the gates in the presence of sufficient fuel.

NAND Gate Reactions: Given a wire identifier w, associate with it the two chemical species $\mathsf{0}_w$ and $\mathsf{1}_w$, representing the value of the wire through the dual-rail convention described earlier. To implement a NAND gate, whose output is 0 if and only if both inputs are 1, with input wires i_1 and i_2 and output wire o, we use the reactions

$$\begin{aligned}
\mathsf{0}_{i_1} + \mathsf{0}_{i_2} + \mathsf{0}_o &\to \mathsf{0}_{i_1} + \mathsf{0}_{i_2} + \mathsf{1}_o,\\
\mathsf{0}_{i_1} + \mathsf{1}_{i_2} + \mathsf{0}_o &\to \mathsf{0}_{i_1} + \mathsf{1}_{i_2} + \mathsf{1}_o,\\
\mathsf{1}_{i_1} + \mathsf{0}_{i_2} + \mathsf{0}_o &\to \mathsf{1}_{i_1} + \mathsf{0}_{i_2} + \mathsf{1}_o,\\
\mathsf{1}_{i_1} + \mathsf{1}_{i_2} + \mathsf{1}_o &\to \mathsf{1}_{i_1} + \mathsf{1}_{i_2} + \mathsf{0}_o.
\end{aligned}$$

In other words, if two inputs "encounter" an output that is erroneous (according to the two inputs), then two inputs cooperate to "fix" the output molecule by converting it to represent the other bit. It is clear that with ideal inputs (i.e., $[\mathsf{0}_i] > 0 \iff [\mathsf{1}_i] = 0$ for $i \in \{i_1, i_2\}$), then at steady state, the output is ideal, correct, and none of the reactions above are possible since all would have at least one reactant with concentration 0. Therefore these reactions are energy-efficient.

By modifying the bits of the rightmost reactant and product in each reaction above to represent a different truth table, the reactions can be made to emulate any 2-input/1-output logic gate.

Fan-out Gate Reactions: A fan-out gate takes a single input wire i and copies its value to two output wires o_1 and o_2. The reactions to implement this are

$$\begin{aligned} \mathsf{0}_i + \mathsf{1}_{o_1} &\to \mathsf{0}_i + \mathsf{0}_{o_1}, & \mathsf{0}_i + \mathsf{1}_{o_2} &\to \mathsf{0}_i + \mathsf{0}_{o_2}, \\ \mathsf{1}_i + \mathsf{0}_{o_1} &\to \mathsf{1}_i + \mathsf{1}_{o_1}, & \mathsf{1}_i + \mathsf{0}_{o_2} &\to \mathsf{1}_i + \mathsf{1}_{o_2}. \end{aligned}$$

As with the NAND gate reactions, it is clear that the reactions are energy-efficient since at steady-state at least one reactant of each reaction has concentration 0.

Signal Restoration Gate Reactions: The NAND and fan-out gates contain no digital abstraction: each input species "pushes" on the output in linear proportion to the concentration of the input. This implies that at best the signal would not be lost; i.e., if input were 90% ideal (e.g. $[\mathsf{0}_i] = 0.9$ and $[\mathsf{1}_i] = 0.1$), then the best the output could be is 90% ideal. In reality, even this is not achieved since our DNA strand displacement implementation of the reactions introduces some non-idealities that imply there will be signal loss at each logical gate. The signal restoration gate described below functions to restore the signal. The gate takes one input i and produces one output o, such that at steady-state, $[\mathsf{1}_o]/[\mathsf{0}_o] = ([\mathsf{1}_i]/[\mathsf{0}_i])^2$. Therefore the difference between the input species that is "high" and the one that is "low" will be amplified.[3]

The following reactions implement this gate. Below we describe the intuition behind them.

$$\mathsf{0}_i + \mathsf{1}_o \to \mathsf{0}_i + \mathsf{1}_o + \mathsf{diff}_o, \tag{1}$$

$$\mathsf{0}_i + \mathsf{diff}_o \to \mathsf{0}_i + \mathsf{p0}_o, \tag{2}$$

$$\mathsf{p0}_o + \mathsf{p0}_o \to \mathsf{p0}_o + \mathsf{p0}_o + \mathsf{P0}_o, \tag{3}$$

[3] The statement "$[\mathsf{1}_o]/[\mathsf{0}_o] = ([\mathsf{1}_i]/[\mathsf{0}_i])^2$ at steady-state" applies to the abstract high-level reactions described in this section but is not entirely accurate for the implementation reactions of Section 2.2 due to the "buffering" effect described in that section. DNA strands spend part of their time "buffered" in double-stranded complexes through reversible exchange reactions even in the absence of the other reactants. The effect is that their "net concentration" may be lower at steady-state by some constant multiplicative factor than their initial concentration (the purpose of the fan-out gates is to keep this factor constant). Nonetheless, we still obtain a quadratic amplification of the ratio $[\mathsf{1}_i]/[\mathsf{0}_i]$ although perhaps off by a constant to account for the buffering effect.

$$\mathsf{P0}_o + \mathsf{1}_o \to \mathsf{P0}_o + \mathsf{0}_o, \tag{4}$$
$$\mathsf{1}_i + \mathsf{0}_o \to \mathsf{1}_i + \mathsf{0}_o + \mathsf{diff}_o, \tag{5}$$
$$\mathsf{1}_i + \mathsf{diff}_o \to \mathsf{1}_i + \mathsf{p1}_o, \tag{6}$$
$$\mathsf{p1}_o + \mathsf{p1}_o \to \mathsf{p1}_o + \mathsf{p1}_o + \mathsf{P1}_o, \tag{7}$$
$$\mathsf{P1}_o + \mathsf{0}_o \to \mathsf{P1}_o + \mathsf{1}_o. \tag{8}$$

Additionally, there is a species decay_o set to some constant concentration (comparable to that of $\mathsf{0}_o$ and $\mathsf{1}_o$), such that, for each species $S \in \{\mathsf{diff}_o, \mathsf{p0}_o, \mathsf{p1}_o, \mathsf{P0}_o, \mathsf{P1}_o\}$, we also have the reaction

$$\mathsf{decay}_o + S \to \mathsf{decay}_o. \tag{9}$$

The intuition behind the reactions above is as follows. Reactions (1) through (4) are functionally the same as (5) through (8); the difference is only in which input bit is being translated to the output. For concreteness we describe only reactions (1) through (4); therefore we regard the input bit as 0, $\mathsf{0}_i$ as the "correct" input species, and $\mathsf{1}_i$ as the "incorrect" input species. Reaction (1) is designed to detect the presence of incorrect output; if an input molecule encounters an output molecule that it "thinks" is incorrect, those molecules catalytically produce a copy of diff_o. The purpose of diff_o is to "announce" to both input species that the output is not ideal; when an input molecule $\mathsf{0}_i$ reacts with diff_o in reaction (2), the input catalytically transforms diff_o into a "push" molecule $\mathsf{p0}_o$.[4] If this molecule $\mathsf{p0}_o$ were to react directly with the output $\mathsf{1}_o$ to convert it to $\mathsf{0}_o$, the rate of conversion would be linear in the input concentration; hence signal restoration would not occur since the reverse conversion of $\mathsf{0}_o$ to $\mathsf{1}_o$ catalyzed by $\mathsf{p1}_o$ would proceed at a rate proportional to the concentration of $\mathsf{1}_i$. To amplify the ratio between the correct input species and the incorrect input species, we must create push molecules whose ratio of correct-to-incorrect is superlinear in the correct-to-incorrect ratio of the input species. This is the function of reaction (3), which produces "strong push" molecules $\mathsf{P0}_o$ at a rate quadratic in the concentration $\mathsf{p0}_o$ (since the rate of a bimolecular reaction of the form $A+A \to \ldots$ is proportional to $[A]^2$). Reaction (4) is then the "correction" of the output species by the strong push molecules.

The reactions are energy-efficient. Although our simulation uses the mass-action kinetics model, it is easiest to describe the intuition with finite counts of molecules, so that molecular concentrations can really go from positive to 0 in a finite amount of time. With ideal input species, say, $[\mathsf{0}_i] > 0$ and $[\mathsf{1}_i] = 0$, reactions (5) and (6) cannot occur. Since (6) cannot occur, production of $\mathsf{p1}_o$

[4] Note that reactions (1) and (5) produce the same molecule diff_o; this is so that, if the output is "close to ideal", the production of output-changing molecules will not become unbalanced in favor of the incorrect output. If reactions (1) and (5) produced molecules specific to one input bit, say $\mathsf{diff0}_o$ and $\mathsf{diff1}_o$, then the production rate of $\mathsf{diff0}_o$ would decrease, and the production rate of $\mathsf{diff1}_o$ would increase, as the output species moved closer to an ideal representation of 0. This would lead to a negative feedback loop hampering signal amplification.

halts and $[\mathsf{p1}_o]$ decays to concentration 0 through reaction (9). With $[\mathsf{p1}_o] = 0$, reaction (7) cannot occur and production of $\mathsf{P1}_o$ halts and $[\mathsf{P1}_o]$ decays to 0 through reaction (9). No reaction at this point can change a $\mathsf{0}_o$ to a $\mathsf{1}_o$. Reactions (1) through (4) continue until all $\mathsf{1}_o$ are converted to $\mathsf{0}_o$. At this point reaction (1) cannot occur, production of diff_o halts, and $[\mathsf{diff}_o]$ decays to concentration 0 through reaction (9). This results in the eventual decay of $\mathsf{p0}_o$ and $\mathsf{P0}_o$, for the same reason as described above for $\mathsf{p1}_o$ and $\mathsf{P1}_o$, at which point none of the reactions (1) through (9) can occur, and the system has reached steady-state.

2.2 Implementation of Abstract Reactions with DNA Strand Displacement

Our design of a NAND gate, a fan-out gate, and a signal restoration gate described in Section 2.1 consists of bimolecular and termolecular reactions (meaning two and three reactants, respectively) with abstract chemicals. We now introduce a method to implement these abstract reactions with DNA strand displacement. For concreteness we show how to implement a bimolecular reaction $A + B \rightarrow C + D$ with two products and a termolecular reaction $A + B + C \rightarrow D + E + F$ with three products. However, it should be clear how to modify each design to allow an arbitrary number of products in either case. For instance, our design from Section 2.1 also requires bimolecular reactions with three products (e.g. $\mathsf{0}_i + \mathsf{1}_o \rightarrow \mathsf{0}_i + \mathsf{1}_o + \mathsf{diff}_o$). As mentioned earlier, this design borrows heavily from [8] but alters the design to allow direct implementation of termolecular reactions. This problem is nontrivial due to the effect of the "buffer" strands for the second reactant, as we will explain below.

The implementation of a bimolecular reaction $A + B \rightarrow C + D$ using DNA strand displacement is shown in Figure 3. The series of reactions is activated in the presence of reactants *ta* (A), *tb* (B) and the gate complex g_1. First, *ta* attaches to the gate g_1 via the uncovered toehold t and the strand displacement follows to detach the buffer1 *at* from g_1 (reaction (a) in Figure 3). At this stage two reactions are possible: the strand displacement caused by *tb* to release the linker (reaction (b)) or the buffer1 strand reverses the previous reaction.Reaction (b) is also reversible due to the exposed toehold on the right side of the complex after reaction (b). At this point the linker can bind to the gate g_2 and release products *tc* (C) and *td* (D) (reaction (c)). Strand 1 and the two complexes with the bottom strands of gates g_1 and g_2 are produced as waste. This reaction is irreversible, but since it is the only irreversible reaction in the cascade, either the entire cascade is "committed" or it is possible to return to the original state shown in the upper left box of Figure 3. In the absence of B, some proportion of copies of strand A spend time "buffered" in gate complex g_1, reversibly exchanging with buffer1, which lowers the "effective concentration" of A, altering the strength of its effect on other reactions of which it is a part. However, given the design of Section 2.1, particularly due to the use of fan-out gates, we may assume each species is a first or second reactant in at most two reactions. Therefore the effective concentration of A is at least a constant fraction of its initial concentration, given a sufficiently large and approximately equal supply of buffer1 and g_1 (in each of the two reactions in which A participates). This is essentially the same argument used in [8] (although handled differently to more precisely simulate desired rate constants, which do not need to be precise for our purposes). As explained below the existence of a second buffer strand in the termolecular reaction requires more sophisticated handling.

The implementation of a termolecular reaction $A + B + C \rightarrow D + E + F$ is illustrated in Figure 4. The essential difference is the buffer2 collector. Whenever the termolecular reaction completes successfully, the numbers of buffer1 and buffer2 increase by 1. Unless processed properly these buffers accumulate and become more and more competitive against A and B. For the buffer1 strand, this problem may be solved by having the gate g_1 and buffer1 be in excess such that their concentrations remain effectively constant; this is the trick used in [8]. However, this argument does not apply to buffer2, since buffer2 reacts with $g_1 : A$ complexes. The number of these is necessarily no larger than the number of A strands, no matter how many gate complexes g_1 are supplied. Therefore the produced copies of buffer2 will eventually grow so large as to effectively prevent C from binding unless the copies of buffer2 are collected. Hence a "collector" is prepared for buffer2, which binds to buffer2 to render it inert. It is critical, however, that the collector not be released but upon the successful completion of the entire cascade of reactions, since we require positive concentration of buffer2 in the case that A and B are present but C is absent, to prevent the undesirable situation that all copies of B become trapped in gate g_1 complexes. Because of the need to deal with buffer1 strands and buffer2 strands differently, it is essential to prevent crosstalk between them; this is explained in Section 2.2.

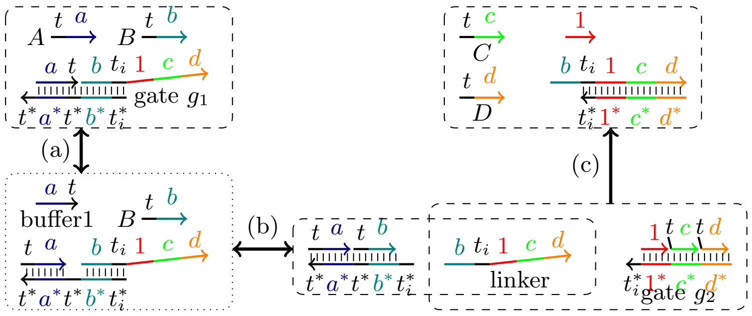


Fig. 3. The DNA motif and reaction mechanism for the bimolecular reaction $A + B \rightarrow C + D$. By $a, b, c, d, 1$, we denote the recognition domains. t and t_i (for $i \in \{1, 2, 3\}$) are the toehold domains. The need for the different toeholds is explained in Section 2.2. The bolded arrow between two dotted squares surrounding respective sets of DNA motifs represents the reaction with the set of DNA motifs at its tail as its reactants and the set at its head as products. Recognition region 1 uniquely identifies this reaction and ensures that the linker strand cannot release any output strands from the gate complex g_2 of other reactions even if they share a first output recognition region (c in this case).

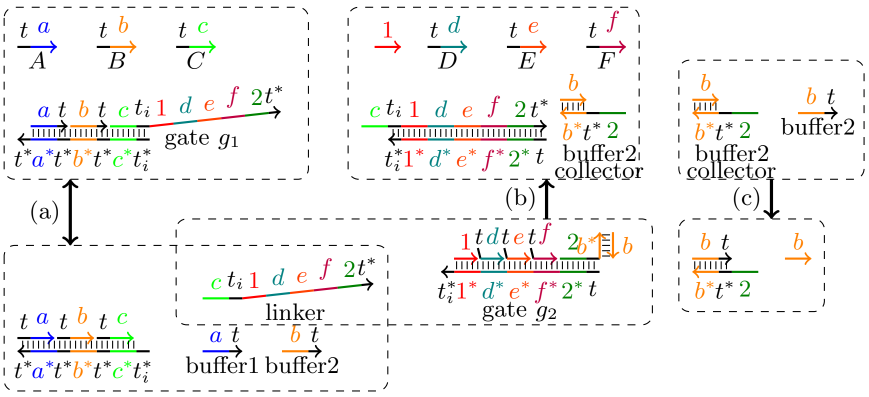


Fig. 4. The DNA motif and reaction mechanism for the termolecular reaction $A + B + C \rightarrow D + E + F$

Our design requires a bit of care with the toeholds. Briefly, each toehold can be equal for any "high-level species strands" (such as A, B, C in Figure 4), and for any buffer strands. However, to avoid unnecessary "crosstalk" between buffers and linkers, and between linkers from two different reactions, it is necessary to use different toehold regions for linkers.

Consider the reactions of Section 2.1. The first three reactions share a third reactant $\mathtt{0}_o$. This implies that the linker strand of Figure 4 corresponding to these reactions shares the first recognition region c representing $\mathtt{0}_o$. If the toehold t_i following the recognition region were equal for these linker strands, then one linker strand could attach to the gate complex g_1 of another reaction, erroneously displacing the strand representing $\mathtt{0}_o$. To prevent this we use different toeholds for each of these reactions that share their third reactant $\mathtt{0}_o$. Similarly, to prevent linkers from displacing non-final reactants (first reactants in a bimolecular reaction, and first or second reactants in a termolecular reaction), we must ensure that $t_i \neq t$ for any $i \in \{1, 2, 3\}$.

This applies to bimolecular high-level reactions as well, but it is routine to verify that no more than three different reactions (the worst case being the case described above) share the same final reactant. It is safe to reuse linker toeholds between reactions that differ in their final reactant because the different initial recognition regions on the linkers will prevent the crosstalk problem described above.

Although we have ensured that buffers and linkers will not have crosstalk because they use different toehold sequences, we must ensure that buffers do not suffer undesired crosstalk with each other. A buffer, unlike a linker, when released uncovers a toehold that is intended to serve as an initial binding site for a strand. Therefore it is not as simple as changing the toehold to prevent buffer crosstalk. Also, a buffer1 strand (a buffer released upon the binding of a first reactant in a reaction, either bimolecular or termolecular) is not garbage-collected as a buffer2 strand is (a buffer released upon the binding of a second reactant in a termolecular reaction). Therefore we must ensure that buffer1 strands are always different from buffer2 strands so that exactly the buffer2 strands are garbage-collected.

To see that this undesired crosstalk does not happen, it suffices to inspect the order of reactants in each reaction. No "high-level species" (corresponding to A, B, etc. in Figures 3 and 4) is a first reactant in one bimolecular or termolecular reaction but a second reactant in another termolecular reactant. Therefore it is critical to maintain the specific order of the reactants we chose for the high-level reactions of Section 2.1.

Acknowledgement. The authors are grateful to Erik Winfree for discussing many issues with us.

References

[1] Goel, A., Ibrahimi, M.: Renewable, time-responsive DNA logic gates for scalable digital circuits. In: Deaton, R., Suyama, A. (eds.) DNA 15. LNCS, vol. 5877, pp. 67–77. Springer, Heidelberg (2009)

[2] Hagiya, M., Yaegashi, S., Takahashi, K.: Computing with hairpins and secondary structures of DNA. In: Nanotechnology: Science and Computation, pp. 293–308 (2006)

[3] Macdonald, J., Li, Y., Sutovic, M., Lederman, H., Pendri, K., Lu, W., Andrews, B.L., Stefanovic, D., Stojanovic, M.N.: Medium scale integration of molecular logic gates in an automaton. Nano Letters 6, 2598–2603 (2006)

[4] Magnasco, M.O.: Chemical kinetics is Turing universal. Physical Review Letters 78(6), 1190–1193 (1997)

[5] Penchovsky, R., Breaker, R.R.: Computational design and experimental validation of oligonucleotide-sensing allosteric ribozymes. Nature 23, 1424–1433 (2005)

[6] Qian, L., Winfree, E.: A simple DNA gate motif for synthesizing large-scale circuits. In: Goel, A., Simmel, F.C., Sosík, P. (eds.) DNA 14. LNCS, vol. 5347, pp. 70–89. Springer, Heidelberg (2009)

[7] Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)

[8] Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. Proceedings of the National Academy of Sciences (March 2010)

[9] Stojanovic, M.N., Mitchell, T.E., Stefanovic, D.: Deoxyribozyme-based logic gates. Journal of the American Chemical Society 124, 3555–3561 (2002)

[10] Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: A DNA-fuelled molecular machine made of DNA. Nature 406(6796), 605–608 (2000)

[11] Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318(5853), 1121–1125 (2007)

Negative Interactions in Irreversible Self-assembly[⋆]

David Doty[1], Lila Kari[1], and Benoît Masson[2]

[1] U. of West. Ontario, Dept. of Computer Science, London, Canada
{ddoty,lila}@csd.uwo.ca
[2] IRISA (INRIA), Campus de Beaulieu, Rennes, France
benoit.masson@irisa.fr

Abstract. This paper explores the use of negative (i.e., repulsive) interactions in the abstract Tile Assembly Model defined by Winfree. Winfree in his Ph.D. thesis postulated negative interactions to be physically plausible, and Reif, Sahu, and Yin studied them in the context of *reversible* attachment operations. We investigate the power of negative interactions with *irreversible* attachments, and we achieve two main results. Our first result is an impossibility theorem: after t steps of assembly, $\Omega(t)$ tiles will be forever bound to an assembly, unable to detach. Thus negative glue strengths do not afford unlimited power to reuse tiles. Our second result is a positive one: we construct a set of tiles that can simulate an s-space-bounded, t-time-bounded Turing machine, while ensuring that no intermediate assembly grows larger than $O(s)$, rather than $O(s \cdot t)$ as required by the standard Turing machine simulation with tiles.

1 Introduction

Tile-based self-assembly is a model of "algorithmic crystal growth" in which square "tiles" represent molecules that bind to each other via highly-specific bonds on their four sides, driven by random mixing in solution but constrained by the local binding rules of the tile bonds. Erik Winfree [10], based on experimental work of Seeman [7], modified Wang's mathematical model of tiling [9] to add a physically plausible mechanism for growth through time. Winfree defined a model of tile-based self-assembly known as the abstract Tile Assembly Model (aTAM). The fundamental components of this model are un-rotatable, but translatable square "tile types" whose sides are labeled with "glues" representing binding sites. Two tiles that are placed next to each other are attracted with strength determined by the glues where they abut, and in the aTAM, a tile *binds* to an assembly if it is attracted on all sides with total strength at least

[⋆] This research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant R2824A01 and the Canada Research Chair Award in Biocomputing to Lila Kari, and by the NSF Computing Innovation Fellowship grant to David Doty.

a certain threshold value τ.[1] Assembly begins from a "seed" tile and progresses until no more tiles may attach.

We study a variant of this model in which glue strengths are allowed to be negative as well as positive. This leads to the situation in which a stable assembly may become unstable through the addition of a tile that, while binding strongly enough to the assembly to remain attached itself, exerts a repulsive force on a neighboring tile, which is sufficiently strong to detach some portion of the assembly. This is formally modeled by allowing an assembly to break into two parts any time that the two parts have total connection strength less than τ (i.e., if there is a cut of the interaction graph of strength less than τ). Negative glue strengths were discussed as a plausible mechanism in Winfree's thesis [10], and explored theoretically in a more general model of *graph-based self-assembly* by Reif, Sahu and Yin [4]. We compare the results of [4] to the present paper in more detail later in this section.

This paper has two main contributions, an impossibility result and a positive result. The impossibility result is that under the irreversible model, negative glue strengths are not sufficient to achieve perfect reuse of tiles as in [4]. It is tempting to believe that with negative glue strengths, the monotonic growth of the aTAM could be overcome to such a degree that a bounded set of tiles could be reused for arbitrarily long computations,[2] hence implementing the observation that "you can reuse space but you can't reuse time". Alas, you cannot reuse space (tiles) too much with irreversible reactions. We show that under the irreversible model of tile assembly, even with negative glue strengths, many tiles will be forever bound to an assembly, unable to detach. In fact, this number is linear in the number of assembly operations, so that after t operations, $\Omega(t)$ tiles will be permanently bound to some assembly.

The positive result is a construction attempting to make do with this limitation. For concreteness, our construction shows how to simulate a single-tape Turing machine. But the idea applies to the iterated computation of any function f that can be "computed with constant height" by a tile assembly system (a formal definition is given in Section 4). The function f_M mapping the configuration of a Turing machine M to its next configuration is an example of one such function. Other examples include the incrementing or decrementing of a counter, or the selection of a uniformly distributed random number from a finite set $\{1, 2, \ldots, n\}$ using flips of a fair coin via von Neumann's *rejection method*, as shown in [2].

Our construction achieves the following property: if the Turing machine M being simulated on input x (with $n = |x|$) has space bound $s(n)$ and time bound $t(n)$, then $O(t(n) \cdot s(n))$ tiles (meaning total count of tiles, which is greater than

[1] The threshold τ models the temperature at which insufficiently strong chemical bonds will break, such as those formed by Watson-Crick complementarity in DNA-based implementations of tiles.

[2] Subject, of course, to computational complexity constraints such as $\mathsf{DTIME}(t(n)) \subseteq \mathsf{DSPACE}(2^{t(n)})$, based on the observation that configurations cannot repeat during the course of a halting computation.

the number of unique tile types), mixed in solution, will simulate the computation of M on input x, and no intermediate assembly will grow to size larger than $O(s(n))$. The impossibility result can be interpreted to imply that external energy must be supplied to break bonds between tiles if we wish to reuse them for computation. If we wish to limit the volume of a solution, and therefore the number of molecules it can contain (by the finite density constraint, see [8]) to $O(s(n))$, then we cannot allow intermediate assemblies to grow larger than this value. Of course, by the impossibility result, many more than $s(n)$ different such assemblies will form if $t(n) \gg s(n)$ (for instance, when simulating a linear-space, cubic-time computation). With a mechanism to "vacuum" away junk assemblies and supply the external energy needed to break them up (a mechanism not modeled in the aTAM), these tiles could be reused, bringing down the required number of tiles from $O(t(n) \cdot s(n))$ to $O(s(n))$.

The main difference between [4] and the present paper is that [4] employs *reversible* reactions, and the present paper employs *irreversible* reactions.[3] Within the aTAM, the main difference between our model and [4] amounts to a difference in the definition of a legal attachment operation. In [4], the authors define a tile attachment to be legal if the tile attaches with strength $\tau - 1$ (in fact, they define it a bit differently but restricting attention to our construction and that of [4], this definition is equivalent). This is a phenomenon not modeled by the aTAM, but it is physically plausible to suppose that it occurs, though with less frequency than strength τ attachments (see the kinetic TAM of [10]). Therefore the tile may detach after attaching since it is held with insufficient strength. But, if it first causes another tile or group of tiles to be bound with total strength less than τ, then those tiles may also fall off, possibly resulting in stabilization of the original attachment. In the present paper, we define attachments to be legal only if they have strength at least τ, whereas detachments may only happen between assemblies attached with strength at most $\tau - 1$. This difference implies that our impossibility result does not apply to [4], which can be considered an advantage of reversible interactions. But this advantage does not come without disadvantages: due to the second law of thermodynamics, their construction must necessarily be implemented as an unbiased random walk with equal rates of forward and reverse reaction, lest the entropy of the system increase with time if one direction is more favorable. Therefore their construction takes expected time n^2 to go forward n steps.

We should also note that although [4] uses a more general model of graph-based self-assembly, this does not imply that their construction of an assembly system simulating a space-bounded Turing machine simulation is a stronger result than our construction. The more general model affords more power to aid in a construction, such as allowing non-planar interactions, in addition to the extra power of reversible interactions. Therefore, we emphasize that our positive construction is not merely a specialization of the construction of [4] to grid graphs. The construction of [4] is inherently non-planar and reversible. A major

[3] [4] also uses a more general graph-based model of self-assembly, but this difference is less crucial than the reversibility issue.

source of the effort in designing our construction was getting it to work in the plane and use irreversible attachments. Many similar (and simpler) constructions that superficially appear to do the same thing as our construction do not actually work in our model, as they introduce not only a desired cut of strength less than τ, but also some undesired cuts of strength less than τ, which, if detached, will ruin the construction.

For color figures, see `http://www.csd.uwo.ca/~ddoty/papers/niisa.pdf`.

2 Abstract Tile Assembly Model

This section gives a brief definition of the abstract Tile Assembly Model (aTAM, [10]) with negative glue strengths. This not a tutorial on the aTAM; for readers unfamiliar with the model, please see [5] for an excellent introduction.

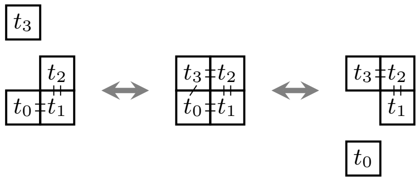
$\mathbb{Z}$ and $\mathbb{Z}^+$ denote the set of integers and positive integers, respectively. Let G be a finite alphabet of *glues*. A *tile type* is a tuple $t \in G^4$, i.e., a unit square with a glue on each side. Associated with the tile types is a *glue strength function* $str : G \times G \to \mathbb{Z}$ that indicates, given two glues g_1 and g_2, the strength $str(g_1, g_2)$ with which they interact. We assume a finite set T of tile types, but an infinite number of copies of each tile type, each copy referred to as a *tile*. Let $G(T)$ denote the set of all glues of tile types in T. An *assembly* (a.k.a., *supertile*) is a positioning of tiles on the integer lattice $\mathbb{Z}^2$ (i.e., a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$, where $\dashrightarrow$ denotes that the function is partial). Each assembly induces a *binding graph*, a grid graph whose vertices are tiles, with an edge between two tiles if they are adjacent (i.e., are Euclidean distance 1 apart).[4] The assembly is τ*-stable*, or simply *stable* if τ is understood from context, if every cut of its binding graph has weight (strength) at least τ, where the weight of an edge is the strength of the glue it represents. That is, the assembly is stable if at least energy τ is required to separate the assembly into two parts. In this paper, where not stated otherwise, we assume that $\tau = 2$.

A *tile assembly system* (TAS) is a 4-tuple $\mathcal{T} = (T, str, \sigma, \tau)$, where T is a finite set of tile types, $str : G(T) \times G(T) \to \mathbb{Z}$ is the *glue strength function*, $\sigma : \mathbb{Z}^2 \dashrightarrow T$ is the finite and τ-stable *seed assembly*, and $\tau \in \mathbb{Z}^+$ is the *temperature*. Given a TAS $\mathcal{T} = (T, str, \sigma, \tau)$, an assembly α is *producible* if either (base case) $\alpha = \sigma$, or (recursive case 1) α results from the τ-stable attachment of a single tile to a producible assembly ("τ-stable attachment" meaning that the cut separating the tile from the rest of the assembly has strength $\geq \tau$), or (recursive case 2) α consists of one side of a cut of strength $< \tau$ of a producible assembly. Note in particular that a producible assembly need not be stable, but may be stabilized by attachments before it can break apart. An assembly α is *terminal* if α is τ-stable and no tile can be τ-stably attached to α. Let $B \subseteq T$ be a set of "black"

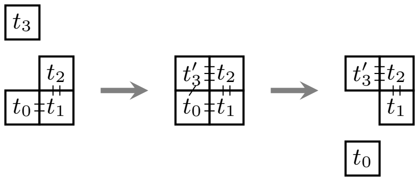
[4] Previous papers model the binding graph as having edges only between tiles that interact with positive strength. In the present paper, the presence of negative glue strengths means that we must consider every possible interaction between adjacent tiles, whether positive, negative, or 0.

tile types. $\mathcal{T}$ is *B-directed* (a.k.a., *B-deterministic*, *B-confluent*) if it has exactly one terminal, producible assembly containing one or more tiles from B.[5]

To define reversible assembly at temperature $\tau = 2$ (as in [4]), it suffices to define attachment events with strength threshold $\tau - 1 = 1$, rather than strength threshold $\tau = 2$. This behavior is illustrated on Fig. 1(a), and can be compared with our new notion, whose evolution is shown on Fig. 1(b).



(a) Reversible model, as defined in [4].



(b) Irreversible model.

Fig. 1. Two different implementations of negative interactions at temperature 2. The slanted bonds represent a strength of -1. In the reversible model, the tile t_3 can attach with a total strength of 1 (one bond of strength 2 and one of strength -1) and hence is unstable, while with our definition, t_3' is attached with a total strength of 2 and forces t_0 to detach.

3 Limitation of Tile Reuse with Irreversible Reactions

If α is an assembly, define $\Phi(\alpha)$, the *(negative) free energy* of α, to be the sum of all glue strengths between adjacent tiles in the assembly.[6] In particular, an assembly consisting of a single tile has free energy 0. If S is a multiset of assemblies (such as that produced by a TAS with negative glue strengths, considering even the "junk" assemblies that are discarded after a cut), define the (negative) free energy of S to be the sum of the free energies of each assembly in S, denoted $\Phi(S)$. Note that even postulating an infinite count of tiles, after a finite number of operations, only finitely many assemblies in S consist of more than one tile, and each of these is a finite assembly. Therefore $\Phi(S) < \infty$ for any multiset S of assemblies producible by a TAS, even in the case that $|S| = \infty$ (such as the initial multiset consisting of a countably infinite number of copies of each individual tile type).

[5] We define this notion of B-directedness but do not henceforth discuss it explicitly, since our construction simulates a general "computation", and B would depend on the goals of the computation being simulated. In our example construction in Section 4 of simulating steps of a Turing machine, B could, for instance, consist of the tile types that represent a halting state, so that only a terminal assembly representing the configuration of a halted Turing machine would be considered the result.

[6] The standard definition of free energy is the negative of this quantity, but as in [5] we use its negation so that the quantity will be positive for stable assemblies. Intuitively, it is the energy *required* to separate α into individual tiles, whereas the standard definition is the energy *released* by such a separation.

When we discuss the "number of steps" for the assembly process of a TAS, we mean the total number of attachment and detachment operations that have been applied so far. We do not claim that this is a proper model of "running time", but it is convenient to think of attachment and detachment events as discrete and equally-spaced steps, even though they may happen in parallel or with interval times governed by a continuous distribution.

Theorem 1. *Let $\mathcal{T}$ be a TAS, and let S be a multiset of assemblies producible by $\mathcal{T}$ after $t \in \mathbb{N}$ steps. Then $\Phi(S) \geq t/2$.*

A proof of Theorem 1 will appear in a full version of this extended abstract.

Since the glue strengths are bounded above by some constant s, an immediate consequence of Theorem 1 is that after t steps, at least $t/(2s)$ sides of tiles are bound. With the finite tile count assumption, once t is sufficiently large that $t/(2s)$ exceeds the total number of sides available (i.e., 4 times the total number of tiles in solution), no more sides are available for binding, and self-assembly grinds to a halt. This is the sense in which a finite number of tiles cannot be reused indefinitely.

There is a natural thermodynamic interpretation of Theorem 1: work done by tiles on tiles, in an irreversible manner, increases the entropy of the system by the second law of thermodynamics, thus decreasing the potential energy available to do more work. Therefore, any potential energy stored in the unattached glues is eventually permanently used up if external energy is not supplied to break these bonds. In our main construction, many junk assemblies are created that are no longer useful once the tiles in them have been used once. Theorem 1 tells us that no amount of cleverness will allow us to break up those assemblies and reuse the tiles solely through design of tile types with negative glues; some external force must be supplied to break them apart using a mechanism not modeled in the aTAM.

Of course, Theorem 1, interpreted in light of the molecular interactions that are being modeled by the aTAM, should not be surprising to any physicist. But we believe it is important to formally establish the truth of such a statement within the model. One develops more confidence in a model of reality when it tells us something already known about reality (e.g., the Positive Mass Theorem [6]).

Theorem 1 does not apply to the negative glue strength construction of Reif, Sahu, and Yin [4], because their model allows *reversible* reactions. Attempting to apply our proof to their model would result in the first inequality $\Phi(S_{i+1}) - \Phi(S_i) \geq \tau$ being replaced by $\Phi(S_{i+1}) - \Phi(S_i) \geq \tau - 1$, which would result in a final lower bound of 0, instead of $t/2$, for $\Phi(S)$. Intuitively, the reversibility of reactions implies that attachment and detachment have symmetric effects on the free energy. But this also implies that their system requires driving the system forward through an unbiased random walk, taking n^2 steps on average to proceed by n net forward steps. Any attempt to speed up the reaction to make the forward rate of reaction faster than the reverse rate of reaction would introduce the imbalance in their respective effects on free energy that allows our proof to work. Therefore this tradeoff in speed versus reusability of tiles is fundamental.

4 Turing Machine Simulation

Throughout this section, fix some finite alphabet Σ. We first describe the class of functions that we will compute, which are intuitively those computable by a constant number of rows of assembly (although the number of columns is unbounded) in the standard aTAM. See [2], for example, for a formal definition of the standard aTAM model. Briefly, it is the same as the model defined in Section 2, but glue strengths are non-negative and are only positive between equal glues.

Definition 1. *Let T be a set of tile types, and let $e : T \to \Sigma$. We say that a row of tiles (a connected subassembly of some assembly of height 1) $t_1, t_2, \ldots, t_k$* e-encodes *a string $x \in \Sigma^k$ if $e(t_1) = x[1], e(t_2) = x[2], \ldots, e(t_k) = x[k]$, where $x[i] \in \Sigma$ is the i^{th} symbol in x. A function $f : \Sigma^* \to \Sigma^*$ is* constant-row computable *if there exists a tile set T, a function $e : T \to \Sigma$, and a constant c such that, for each $x \in \Sigma^*$, there is a height-1 stable assembly $\sigma_x : \mathbb{Z}^2 \dashrightarrow T$ e-encoding x such that the tile assembly system $\mathcal{T} = (T, str, \sigma_x, 2)$ (with $str(g_1, g_2) > 0 \iff g_1 = g_2$) has the unique terminal assembly α, the height of α is c, the bottom row of α is σ_x, the top row of α e-encodes $f(x)$, and the leftmost column of any row of α is no further left than the bottom row.*

The widths of the rows representing the input and output may be different (i.e., possibly $|x| \neq |f(x)|$). In this case, we require only that the leftmost and rightmost tiles of each row have their glues specially marked to distinguish them from the tiles interior to the row.

Our construction shows how to design a tile set that will compute iterations of any constant-row computable function f, ensuring that no intermediate assembly grows larger than the size of the input or output processed by any *individual* invocation of f. Examples of such functions include the function f that, given a configuration of a single-tape Turing machine outputs the next configuration of this Turing machine, or that increments a counter represented in binary.

Figure 2 shows a high-level overview of the entire construction, in terms of a general constant-row computable function f. For concreteness, think of f as the function that, given a configuration of a t-time-bounded, s-space-bounded, single-tape Turing machine, outputs the next configuration of this Turing machine (extending the tape on the right side only). The construction proceeds as follows, each label corresponds to a picture in Figure 2.

(a) First, the scaffold tiles (green) connect to the x data assembly (white). The scaffold tiles initiate the computation of f (blue).

(b) The scaffold "detects" when the computation is finished, in the sense that the green row above $f(x)$ tiles cannot complete until all of $f(x)$ is present. Then the scaffold tiles grow back to the first scaffold tile to initiate the removal of $f(x)$ from the tiles surrounding $f(x)$.

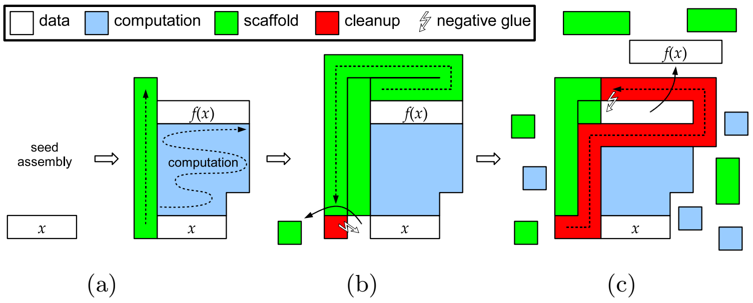


Fig. 2. High-level overview of assembly for computation of a constant-row computable function f

(c) The removal tiles (red) each use a negative glue strength against the tile "in front of" (on the path show by the arrows) it, and once this tile is removed, a new removal tile grows in its place to continue the removal. The path and bond placements and strengths are carefully chosen to ensure that no portion of $f(x)$ is removed, until the last step when $f(x)$ detaches whole from the rest of the tiles.

Note that since f is constant-row computable, the height of the scaffold and removal parts are bounded by a constant and therefore may be hard-coded into the tile set, whereas special glues mark the horizontal endpoints so that the length of x and $f(x)$ are not constrained.

The simulation of the Turing machine for t steps will then consist of executing this assembly process for t iterations, using the output assembly $f(x)$ as the input assembly x for the next execution. After each iteration, the width of the remaining "junk" assembly is a constant plus $O(1) + \max\{|x|, |f(x)|\}$, and the height is constant since f is constant-row computable, so the size of the intermediate assemblies is $O(\max\{|x|, |f(x)|\})$.

Figures 3, 4, and 5 give some more details for the three main steps of Figure 2, respectively (a), (b), and (c), using the specific example of f mapping a configuration of a single-tape Turing machine to its next configuration.

Figure 3 shows an example of tiles implementing step (a) of Figure 2, i.e., the computation of f. The example shows one transition of a single-tape Turing machine, with tape contents 01_0 (_ standing for blank), in state q, with tape head on the rightmost cell, transitioning to state p, moving the tape head right, changing the cell's symbol from 0 to 1, and encountering a blank on the new rightmost cell. In this case, a new rightmost cell is needed, illustrating how our construction handles dynamically changing space requirements, but if the tape head were further left in the row, it would simply fill in copy tiles to the right, just as to the left as shown above, and the row would stay the same width. At the start and end of a computation, the configuration is copied so that any strength > 1 bonds used in the computation are on the interior of the computation tiles, ensuring that only strength-1 bonds must later be broken to separate the data

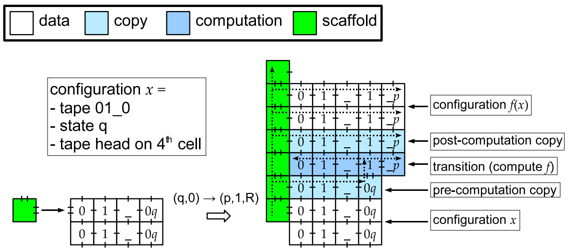


Fig. 3. Example of tiles implementing the computation step. Arrows within tiles show order of growth. In this case f is constant-row computable with constant $c = 1$. The first and last copy rows, shown in lighter shade than the center computation tiles, are always present no matter the function f, and their placement is initiated by the scaffold tiles. However, there is no interaction between the center computation and scaffold tiles. Note that the data tiles are two rows with strength 1 glues; this is to make them stable at temperature 2 but not producible (without additional scaffolding) as they would be if they were a single row connected with strength 2 glues.

tiles. Each data assembly on either end of the computation tiles is represented by a two-row assembly with only single-strength bonds on its interior, which ensures that when detached, the data assembly will be stable, but that it cannot form on its own without help from the scaffold tiles (which would happen if it were only a single row connected with strength-2 bonds). Each vertical position is hard-coded into the tile set; i.e., the scaffold tile set "knows" the required height to compute f. However, the absolute horizontal positions are not encoded into the tiles, only the leftmost and rightmost tiles of the configuration are specially marked, and all interior tiles representing the same data are identical.

Figure 4 shows the tiles implementing step (b) of Figure 2, positioning the tiles for cleanup. The top two rows must use cooperation to tell where the end of the

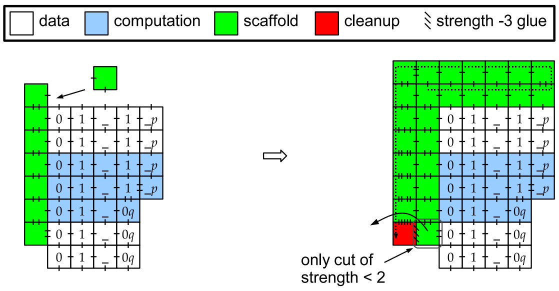


Fig. 4. Tiles that position the cleanup tiles. Here the "copy" tiles from Figure 3 are depicted in the same shade as the computation tiles; now that $f(x)$ has been computed our goal is to remove all of them from the subassembly representing $f(x)$. The order of growth of the scaffold tiles ensures that cleanup does not begin until all of $f(x)$ is present.

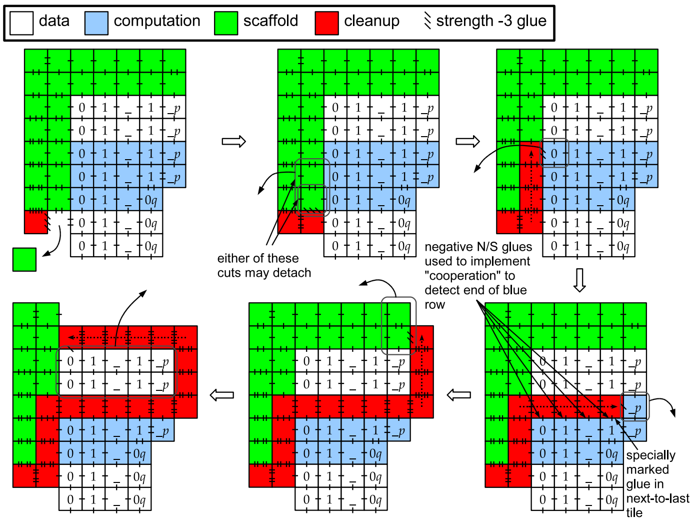

data
computation
scaffold
cleanup
strength -3 glue
either of these cuts may detach
negative N/S glues used to implement "cooperation" to detect end of blue row
specially marked glue in next-to-last tile

Fig. 5. Tiles that "clean up" the connections between the output data and the scaffold and computation tiles to separate them and allow the data tiles to be computed on again

row underneath is, since the width of the output row is unknown. The strengths of bonds on the leftmost downward-growing column must be sufficiently large to ensure that only the proper cut is made when the first negative-strength glue is applied.

Figure 5 shows the tiles implementing step (c) of Figure 2, "cleaning up" by removing the output $f(x)$ from the scaffold, computation, and x tiles. Though not shown, negative strength interactions are necessary between the second-to-top row of computation tiles and some of the right-growing cleanup tiles, to ensure that the right end of the row is properly detected. That is, there are two types of cleanup tiles growing right, one to detach the interior tiles, and one to detach the final rightmost computation tile. Since the east-west bonds between cleanup tiles are greater than 1, the negative north-south glue strengths between interior cleanup tiles and the second-to-rightmost blue tile – and between the rightmost cleanup tile and the interior computation tiles – must be strength -2 to ensure that the second-to-rightmost blue tile cannot stably attach except where intended.

5 Conclusion

We have shown two main results in the aTAM with negative glue strengths, under the standard assumption of *irreversible* attachment, meaning attachments that only occur with strength at least the temperature τ. The first result is that the amount of tile reuse afforded by the ability to detach tiles with negative glue strengths is fundamentally limited. After t steps of assembly, $\Omega(t)$ tiles are permanently bound, unable to detach via negative glue strengths, and can only be detached by supplying external energy. The second result is a positive result that attempts to make do with this limitation: an $s(n)$-space-bounded Turing machine may be simulated for arbitrarily many steps, while ensuring that no intermediate assembly grows larger than $O(s(n))$.

Space-bounded "computation" as an end goal is not the only application of negative glue strengths, of course. Doty, Lutz, Patitz, Summers, and Woods [2] study the problem of generating uniform random distributions on the finite sets using the independent flips of a fair coin afforded by the random selection of competing tile types in the aTAM (a non-trivial problem when the cardinality of the set is not a power of the number of competing tile types), and find a tradeoff between the closeness to uniformity of the distribution obtained and the space required for sampling. They exhibit a construction imposing a perfectly uniformly distribution on the set $\{0, 1, \ldots, n-1\}$ that assembles a structure of width $\lfloor \log n \rfloor + 1$ and *expected* height at most 2, essentially implementing von Neumann's *rejection method* of flipping $\lfloor \log n \rfloor + 1$ fair coins repeatedly and stopping the first time that they encode a number smaller than n. It is very unlikely (probability at most 2^{-20}) to take more than 20 attempts. But using this method in a construction such as that of [3],[7] in which many (perhaps more than 2^{20}) copies of this experiment repeat throughout assembly, could increase the likelihood of growing too high. Even a single occurrence of a too-high subassembly will destroy the entire construction. Though we omit the details in the extended abstract, it is straightforward to augment the construction of [3] (which uses a variant of the random number selector of [2]) with negative glue strengths to implement perfectly uniform selection of random numbers, thus improving the fidelity of the simulation of [3], while providing an absolute guarantee on the space bound.

There are other uses of negative glues in the aTAM. For instance, we are able to improve the best known tile complexity (number of tile types) required to uniquely assemble a "thin" rectangle, i.e., an $n \times k$ rectangle with $k < \log n / (\log \log n - \log \log \log n)$. In the standard aTAM the tile complexity of this shape is known to be $\Omega(\frac{n^{1/k}}{k})$ [1]. With the model of negative glue strengths we are able to improve this to $O\left(\sqrt{\log n}\right)$, by first building a thick rectangle and using negative glues to "cut out" a thinner rectangle of the same length.

Other questions related to this work include the experimental aspects of such a model, for example, how repulsive forces can be realized on DNA tiles, and how to "clean" and "recycle" the junk introduced during the assembly.

[7] The main construction of [3] shows how a "universal" tile set can be constructed that can be "programmed" through appropriate selection of a seed assembly to simulate the growth of any tile assembly system in a wide class of systems termed "locally consistent" (see [3] for details). In this discussion, we are concerned only with the fact that the construction of [3] 1) requires random numbers to be generated in a bounded space at many points throughout assembly, and 2) would be improved if the distribution of these numbers were perfectly uniform instead of "close to uniform" as in [3].

Acknowledgments. The authors are grateful to Shinnosuke Seki for insightful discussions and anonymous referees for their suggestions.

References

[1] Aggarwal, G., Cheng, Q., Goldwasser, M.H., Kao, M.-Y., Moisset de Espanés, P., Schweller, R.T.: Complexities for generalized models of self-assembly. SIAM Journal on Computing 34, 1493–1515 (2005); Preliminary version appeared in SODA 2004

[2] Doty, D., Lutz, J.H., Patitz, M.J., Summers, S.M., Woods, D.: Random number selection in self-assembly. In: Calude, C.S., Costa, J.F., Dershowitz, N., Freire, E., Rozenberg, G. (eds.) UC 2009. LNCS, vol. 5715, pp. 143–157. Springer, Heidelberg (2009)

[3] Doty, D., Lutz, J.H., Patitz, M.J., Summers, S.M., Woods, D.: Intrinsic universality in self-assembly. In: STACS 2010: Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science. Leibniz International Proceedings in Informatics (LIPIcs), vol. 5, pp. 275–286 (2010)

[4] Reif, J.H., Sahu, S., Yin, P.: Complexity of graph self-assembly in accretive systems and self-destructible systems. In: Carbone, A., Pierce, N.A. (eds.) DNA 11. LNCS, vol. 3892, pp. 257–274. Springer, Heidelberg (2006)

[5] Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: STOC 2000: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pp. 459–468 (2000)

[6] Schoen, R., Yau, S.-T.: On the positive mass conjecture in general relativity. Communications in Mathematical Physics 65(45), 45–76 (1979)

[7] Seeman, N.C.: Nucleic-acid junctions and lattices. Journal of Theoretical Biology 99, 237–247 (1982)

[8] Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Natural Computing 7(4), 615–633 (2008)

[9] Wang, H.: Proving theorems by pattern recognition – II. The Bell System Technical Journal XL(1), 1–41 (1961)

[10] Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology (June 1998)

Search and Validation of Short Genome-Wide Biomarkers for Bacterial Biological Phylogenies

Max H. Garzon[1] and Tit-Yee Wong[2]

[1] Computer Science, The University of Memphis, Tennessee 38152
[2] Biology, The University of Memphis, Tennessee 38152
{mgarzon,tywong}@memphis.edu

Abstract. We continue the exploration of DNA-based indexing as a universal coordinate system in DNA spaces to characterize very large groups (families, genera, and even phylla) of organisms on a uniform biomarker reference system, a comprehensive "Atlas of Life", as it is or as it could be on earth. We provide a second confirmation that DNA noncrosshybridizing (nxh) sets can be successfully applied to infer *ab-initio* phylogenetic trees by providing a method to measure distances among entire genomes indexed by sets of short oligonucleotides selected so as to minimize crosshybridization. These phylogenies are solidly established and well accepted in bacterial biology, albeit done by analyses of relatively small segments of highly conserved rybozomic DNA. Second, it is further demonstrated that DNA indexing does provide novel and principled genome-wide predictions into the phylogenesis of organisms hitherto inaccessible by current methods, such as a prediction of the origin of the Salmonella plasmid 50 as being acquired horizontally, likely from some bacteria somewhat related to Yesinia. We conclude with some discussion about the scalability and potential of this method to develop a comprehensive tree of life based on genome-wide methods.

Keywords: DNA codeword design, noncrosshybridizing oligonucleotide bases, phylogenetic analysis, genomic signatures, DNA chips, 16S rRNA tree of life.

1 Introduction

DNA computing (Garzon and Yan, 2008; Adleman, 1994) has originated novel ideas for a variety of techniques aimed at finding large sets of short oligonucleotides with noncrosshybridizing (nxh) properties by several groups, particularly the PCR Selection (PCRS) protocol used below (Garzon et al., 2009; Deaton et al., 2006; Tulpan et al., 2005; Chen et al., 2006). In previous work, these sets have found new applications, such as novel DNA approaches to natural language processing (Bobba et al., 2006) and DNA-based memories (Neel and Garzon, 2006), and more recently, biological phylogenies based purely on genomic DNA (Garzon et al., 2009), where a review of pylogenetic methods can be found. The foundations of the method developed in prior work are summarized in Section 2.1. Section 2.2 reviews the new method for phylogenies from genome-wide data alone, including its preliminary validation in reproducing *ab initio* a well established and accepted phylogeny in biology, the 16S rRNA tree of life, from genomic data alone. In Section 3, we show how these methods reveal new insights into the phylogenesis of organisms hitherto inaccessible by current methods, such as a prediction of the origin of the Salmonella plasmid 50 as being acquired horizontally, likely from some bacteria somewhat related to Yesinia. We conclude with some discussion in Section 4 about the scalability and potential of this method to develop a comprehensive tree of life based on genome-wide methods.

2 Genome-Wide Methods in Phylogenesis

In this section we briefly review previous work on genome-wide methods for phylogenetics to situate the main results in Section 3. We focus on the aspects relevant to the validation described later in Section 3. A more extended review of biological phylogenies can be found in (Garzon et al., 2009).

2.1 Biomarkers for Phylogeny

DNA barcoding has been used as a technique for characterizing species of organisms using a short DNA sequence from a standard and agreed-upon position in the genome.

DNA barcode sequences are very short relative to the entire genome and they can be obtained reasonably quickly and cheaply. For example, "the cytochrome c oxidase subunit 1 mitochondrial region (COI)" is a highly conserved gene in the animal kingdom. The Barcode Initiative aims to construct a public reference library of species identifiers which could be used to assign unknown specimens to known species. Nearly optimal theoretical and practical algorithms have been developed to barcode given genome families (Zhou et al., 2008).

As useful as it might appear for phylogenetic purposes, identifying organisms based on a single marker is far from ideal for the purposes of phylogenetics. What is desirable is a more comprehensive set of features describing the biological functions that characterize individuals in a given set of organisms. Although there are several biomarkers commonly used in phylogenetic studies, not all of these biomarkers are present in all living cells (for example, most bacteria do not have cytochrome c), making using cytochrome c sequence as a reference for the tree of life difficult. Also, it is difficult to know the natural history of the genes selected for biomarkers. As a result, biomarkers based on larger genomic segments are desirable, particularly in view of the rapid accumulation of complete sequences of a number of prokaryotic genomes that has made it possible to analyze the relationships between organisms at the whole-genome level. However, most of the approaches are still relatively subjective and far from being a universal barcode representing *all genes of a species for all species*.

Ideally, an organism should be defined by all the genes in the genome of that species, not by a single gene (monophasic), nor by a few selected genes (polyphasic.) To this end, a new technique was introduced in (Garzon et al, 2009). In this section we briefly summarize these results in order to set up the validation of the scalability of these results described in Section 3.

2.2 Phylogeny by Noncrosshybridizing Sets

The general foundations of the genome-wide phylogenetic methods have been described in (Garzon et al., 2009; Bobba et al., 2006). The basic idea is to use so–called noncrosshybridizing (nxh) sets of probes affixed on a DNA chip. In practice, such a set is a judicious selection of (the complements) of some fragments of some target gene(s) from a target organism, or even a full selection of the target organism (as in the case of a DNA microarray.) A given (possibly unknown) target is digested, usually tagged, and poured over the chip. A signal is produced by a pattern of hybridization to the probes on the chip. If the oligo probes on the chip are not pre-processed, the result will be a signal that can be highly variable and essentially unreproducible upon repetition of the readout. On a noncrosshybridizing chip, a random target digested to fragments of comparable probe size is much more likely to hybridize to fewer probes, and under appropriate stringency ρ, ***to at most one probe***. This so-called *nxh property* immediately translates into the desirable properties to the problems mentioned above. The noise is notably reduced (in fact, it is completely eliminated under ideal conditions), results will be more predictable, and the corresponding analyses will be much more reliable, as argued in (Garzon et al., 2009). The original proof of concept of the nxh method was developed based on the set of bacteria described in Table 1.

The new genomic signatures described below (Figs. 1, 4) have been obtained using a simulation tool, a virtual test tube, *Edna*, developed by the first author in the last few years for simulating the DNA reactions such as hybridization, ligation, self-assembly, and enzymatic reactions (Garzon et al., 2005; Blain et al., 2004). *Edna* has produced results highly correlated, if not identical, to the results of experiments *in vitro*, including a successful run of Adleman's solution to the difficult Hamiltonian Path Problem (Adleman, 1994) *in silico* for graphs up to 12 vertices producing error-free solutions (Garzon et al., 2004). Therefore, there is good evidence that Edna will produce high reliable estimates of the DNA reactions and other events in a wet tube.

Table 1. A selection of six (6) pervasive and important bacterial organisms selected for evaluation of the new phylogenetic analysis described in Section 3. Their nucleic acid files were downloaded in FASTA form and a phylogenetic analysis made to produce a tree, using JavaTreeView, based on a refinement of the 16S rRNA tree of life, originally constructed on the analysis of highly conserved sequences by traditional methods in phylogenetic analysis. (Garzon et al., 2009).

Organism	Genome Size (Mbytes) / ORFs selected (Kbytes)
Escherichia coli CFT073	5.05Mb / 510K
Escherichia coli K12	4.37Mb / 438K
Photobacterium Profundum	3.59Mb / 186K
Pseudomonas Aeruginosa PA01	5.89Mb / 192K
Salmonella Enterica Choleraes.	4.51Mb / 151K
Vibrio. Fisheri ES114	2.64Mb / 179K

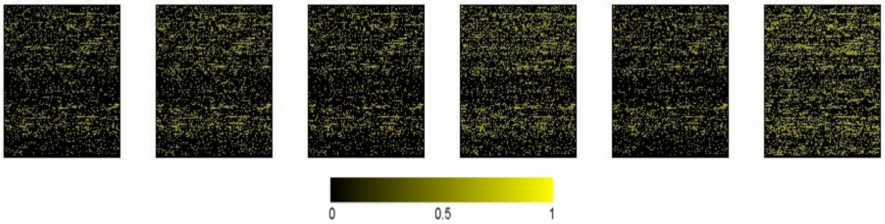
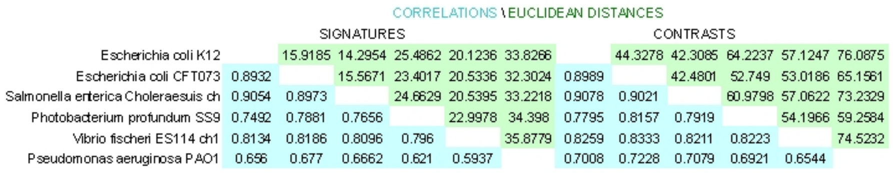


Fig. 1. Genomic signatures of six bacteria (permuted from Table 1) on a nxh DNA chip obtained by PCR Selection from a seed pool of shredded ORFs of their chromosomes. The signatures show clearly significant relative differences and can be contrasted by objective measures to build a phylogenetic tree. Furthermore, probes with most significant similarities and differences can be extracted by pixelwise comparison of pairs of genomic signatures. (Garzon et al., 2009).

For the evaluation of this method, we used an expansion of this generally accepted scheme, the 16S rRNA tree (a fragment is shown in Fig. 3, left), in order to compare organisms outside the region of highly conserved genes (Wong et al., 2009). Using as a seed pool the same selection of ORFs from these organisms used to obtain the extended 16S rRNA tree (Table 1), the PCR Selection protocol was run in simulation to obtain a chip of about 1600 16-mers for these targets by the methodology described in (Garzon et al., 2009; Deaton et al., 2006; Chen et al., 2006). Fig. 1 shows the resulting signatures of the bacteria given in Table 1 reported in (Garzon et al., 2009). The signatures show a clear difference between the target genomes, not only visually to the human eye, but through more objective measures. The obvious measure, the plain Euclidean distance between signatures vectors in Euclidean space (upper triangles), already shows a high degree of correlation comparable to the 16S rRNA tree. Nonetheless, we used a more refined measure in our analysis and produced a phylogenetic tree using identical methods to the ones for the 16 rRNA tree described above. The new metric is the *contrast* between two signatures *x* and *y* given by

$$Co(x,y) := average_k \{c_k\} / std_k \{c_k\}, \text{ where } c_k := average_i \{| x_k - y_i |\},$$

where std_k is the standard deviation of signature over all probes *k*.



CORRELATIONS \ EUCLIDEAN DISTANCES

	SIGNATURES						CONTRASTS					
Escherichia coli K12		15.9185	14.2954	25.4862	20.1236	33.8266		44.3278	42.3085	64.2237	57.1247	76.0875
Escherichia coli CFT073	0.8932		15.5671	23.4017	20.5336	32.3024	0.8989		42.4801	52.749	53.0186	65.1561
Salmonella enterica Choleraesuis ch	0.9054	0.8973		24.6629	20.5395	33.2218	0.9078	0.9021		60.9798	57.0622	73.2329
Photobacterium profundum SS9	0.7492	0.7881	0.7656		22.9978	34.398	0.7795	0.8157	0.7919		54.1966	59.2584
Vibrio fischeri ES114 ch1	0.8134	0.8186	0.8096	0.796		35.8779	0.8259	0.8333	0.8211	0.8223		74.5232
Pseudomonas aeruginosa PAO1	0.656	0.677	0.6662	0.621	0.5937		0.7008	0.7228	0.7079	0.6921	0.6544	

Fig. 2. Correlations (lower triangles) and Euclidean distances (upper triangles) between six bacterial signatures (column 1) on the DNA chip described in Fig. 1. They give rise to a contrast-based phylogenetic tree **T16 cDNA** that essentially reproduces the corresponding fragment of an expanded generally accepted tree **16S rRNA.** (Garzon et al., 2009).

This new phylogenetic distance *Co(x,y)* between genomes *x,y* measures the degree of similarity between two organisms by the contrast $\{c_k\}$ (normalized) signatures, given by the average value of the Signal-to-Noise Ratio (SNR) in the entries in the contrast matrix of their genomic signatures *x* and *y*. The results are shown in Fig. 2. Based on this new contrast metric, a corresponding phylogenetic tree **16 cDNA** was obtained as shown in Fig. 3 (right). As can be observed in the differences (right column), the 16 cDNA tree is identical to the original 16S sRNA tree described above, except for a permutation of two contiguous genomes. It is remarkable that this result has been obtained purely on genomic analysis, independently of the many other considerations that led to the original tree of life 16 sRNA.

An additional consequence is worth pointing out. These results also provide evidence to counter the superficial observation that genomic signatures cannot provide an adequate tool for genetic comparison since they ignore the transcription process altogether (and so do not involve protein or protein expression directly.) 16S rRNA is essentially protein-based and yet, the phylogenetic tree obtained by the contrast metric from Fig. 2 shows that it is well approximated by analyses of genomic signatures alone on a 16-mer nxh DNA chip built *a priori* and designed for the collection of 6 subfamilies of plasmids and bacteria used in this original and preliminary validation.

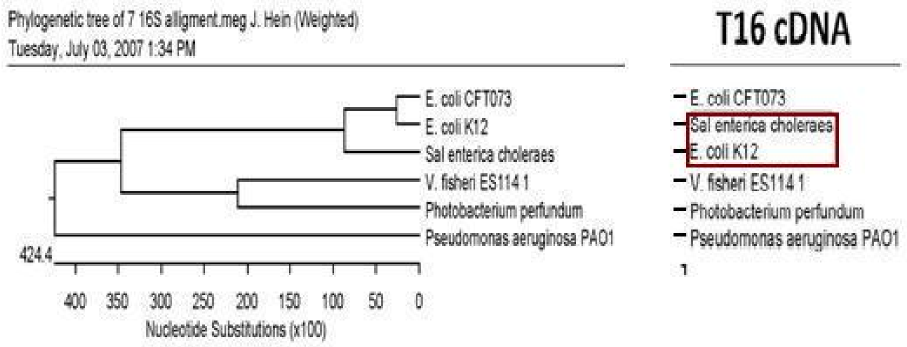


Fig. 3. 16S rRNA (left): Phylogenetic tree of six (6) bacterial chromosomes shown in Fig. 1. The tree **T16 cDNA** (right) is identical to it, except for one difference (transposition of *Sal. Enterica and E-Coli K12*), which is thus *essentially reproduced ab initio* by comparative genomic analyses of their (digital) genomic signatures (shown in Fig. 2) on a DNA chip of noncrosshybridizing (nxh) 16-mer probes (Garzon et al., 2009).

3 Scalability of Genomic Signatures

These foregoing preliminary results posed the question of their scalability to larger phylogenies. In a much larger study, the ORFs of the organisms in Table 2 were processed in a similar manner. This set included the *Escherichia-Shigella-Salmonella* group associated with another entric bacteria *Yesinia pestis*, the causing agent of the Black Death that ravaged Europe in the middle ages. It has been proposed that members of this group were spited as recently as 150 million years ago (Ochman, Elwyn et al. 1999). The exact origin of *Yesinia* is still unclear. Also included is another interesting group of bacteria with unclear origin, the *Neisseria.* This family is commensal bacteria that colonize the mucosal surfaces of many higher animals. Of the eleven species that colonize humans, only two are pathogens. *N. meningitidis* and *N. gonorrhoeae* often cause asymptomatic infections. On the other hand, the *Rickettsia* is obligate parasite carried by ticks, fleas and lice, and cause diseases including typhus and rocky mountain spotted fever in humans. The *Pseudomonas* is a loosely grouped familiy. All of them are gram negative, free-living, aerobic bacteria that exhibit diverse metabolic activities. They were included in an attempt to test whether of the T16 cDNA method could produce a reasonable guess at their (unknown) natural phylogeny. In addition to their major circular chromosome, bacteria often contain various types of extrachromosomal DNA in the forms of minor chromosomes, magaplasmids, and plasmids, the origins of which are hard to predict. It has been hypothesized that some of these extrachromosomal DNA were the results of uneven DNA replication, and some of them were the result of horizontal gene

Table 2. An expanded selection of pervasive and medically important bacterial genomes for evaluation of the new pylogenetic analysis described in this Section. Their nucleic acid files were selected and processed as described above for the preliminary set in Table 1.

Organism	Genome Size
Escherichia coli CFT073	5.05Mb
Escherichia coli K12	4.37Mb
Escherichia coli O15-7-H7 VT2Sakai	5.6 Mb
Neisseria gonorrhoeae FA1090 (Oklahoma)	2.15 Mb
Neisseria meningitidis FAM18	2.2 Mb
Pseudomonas fluorescens Pf-5	7.1 Mb
Pseudomonas entomophila L48	5.9 Mb
Pseudomonas aeruginosa PA01	5.89Mb
Rickettsia felis URRWXCal2	1.59 Mb
Rickettsia conorii Malish 7	1.27 Mb
Salmonella enterica Paratyphi ATCC9150	4.59 Mb
Salmonella typhimurium LT2 SGSC1412	4.99 Mb
Salmonella Enterica Choleraes Plasmid 50	0.42 Kb
Shigella dysenteriae Sd197	4.56 Mb
Shigella boydii Sb227	4.63 Mb
Yersinia pestis KIM	4.7 Mb
Yersinia pestis Antiqua	4.88 Mb

transfers (Volff and Altenbuchner 2000). These extrachromosomal DNAs often contain genes that, although inessential to the their host, could enhance the host's survival by helping produce resistance to antibiotics, for example. *Salmonella enterica* was also added to test whether this whole genome approach could identify the origin of one interesting plasmid (Pl 50) in the *Salmonella Enterica Choleraes* commonly found in pigs.

Using the nxh method, a highly refined set of 27 nxh probes was obtained by PCR Selection for the bacteria in Table 2. The results are shown in the corresponding Figs. 4, 5, and 6.

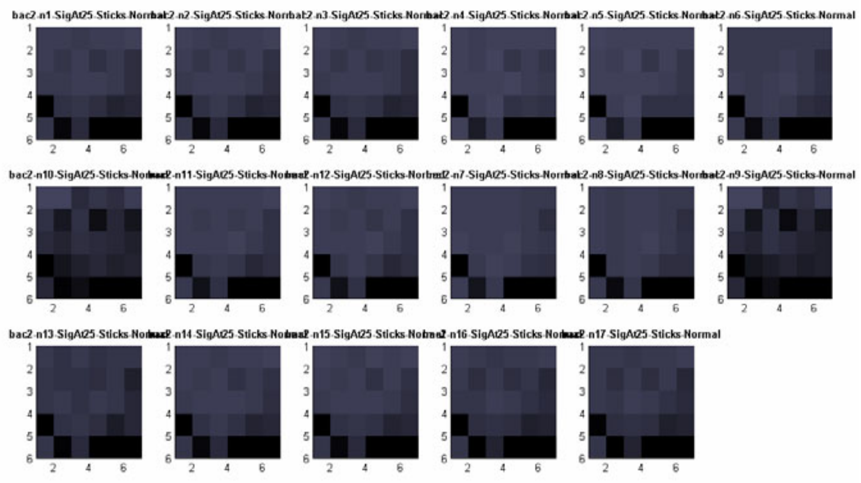
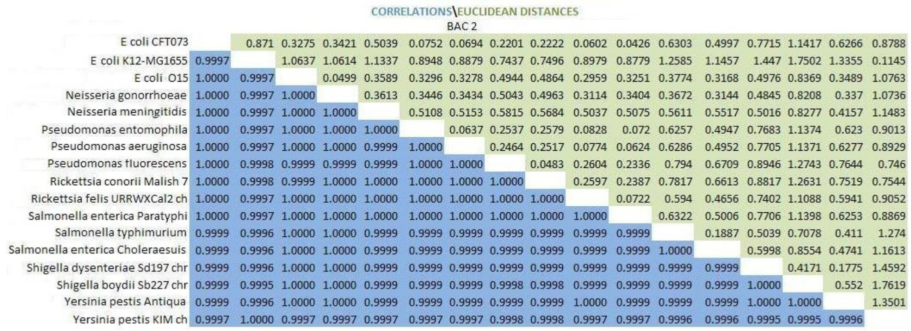


Fig. 4. Genomic signatures of seventeen (17) bacteria in Table 2 (permuted) on a nxh DNA chip (27 highly selective probes of length between 9- and 20-mers) obtained by PCR Selection from a seed pool of shredded ORFs of their chromosomes, analogous to the protocol used in the preliminary data set. The signatures again reveal significant relative differences and can be contrasted by objective measures to build a phylogenetic tree.



CORRELATIONS\EUCLIDEAN DISTANCES

BAC 2

E coli CFT073		0.871	0.3275	0.3421	0.5039	0.0752	0.0694	0.2201	0.2222	0.0602	0.0426	0.6303	0.4997	0.7715	1.1417	0.6266	0.8788
E coli K12-MG1655	0.9997		1.0637	1.0614	1.1337	0.8948	0.8879	0.7437	0.7496	0.8979	0.8779	1.2585	1.1457	1.447	1.7502	1.3355	0.1145
E coli O15	1.0000	0.9997		0.0499	0.3589	0.3296	0.3278	0.4944	0.4864	0.2959	0.3251	0.3774	0.3168	0.4976	0.8369	0.3489	1.0763
Neisseria gonorrhoeae	1.0000	0.9997	1.0000		0.3613	0.3446	0.3434	0.5043	0.4963	0.3114	0.3404	0.3672	0.3144	0.4845	0.8208	0.337	1.0736
Neisseria meningitidis	1.0000	0.9997	1.0000	1.0000		0.5108	0.5153	0.5815	0.5684	0.5037	0.5075	0.5611	0.5517	0.5016	0.8277	0.4157	1.1483
Pseudomonas entomophila	1.0000	0.9997	1.0000	1.0000	1.0000		0.0637	0.2537	0.2579	0.0828	0.072	0.6257	0.4947	0.7683	1.1374	0.623	0.9013
Pseudomonas aeruginosa	1.0000	0.9997	1.0000	1.0000	0.9999	1.0000		0.2464	0.2517	0.0774	0.0624	0.6286	0.4952	0.7705	1.1371	0.6277	0.8929
Pseudomonas fluorescens	1.0000	0.9998	0.9999	0.9999	0.9999	1.0000	1.0000		0.0483	0.2604	0.2336	0.794	0.6709	0.8946	1.2743	0.7644	0.746
Rickettsia conorii Malish 7	1.0000	0.9998	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000		0.2597	0.2387	0.7817	0.6613	0.8817	1.2631	0.7519	0.7544
Rickettsia felis URRWXCal2 ch	1.0000	0.9997	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000		0.0722	0.594	0.4656	0.7402	1.1088	0.5941	0.9052
Salmonella enterica Paratyphi	1.0000	0.9997	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000		0.6322	0.5006	0.7706	1.1398	0.6253	0.8869
Salmonella typhimurium	0.9999	0.9996	1.0000	1.0000	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999		0.1887	0.5039	0.7078	0.411	1.274
Salmonella enterica Choleraesuis	0.9999	0.9996	1.0000	1.0000	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	1.0000		0.5998	0.8554	0.4741	1.1613
Shigella dysenteriae Sd197 chr	0.9999	0.9996	1.0000	1.0000	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999		0.4171	0.1775	1.4592
Shigella boydii Sb227 chr	0.9999	0.9995	1.0000	1.0000	0.9999	0.9999	0.9999	0.9998	0.9998	0.9999	0.9999	0.9999	0.9999	1.0000		0.552	1.7619
Yersinia pestis Antiqua	0.9999	0.9996	1.0000	1.0000	0.9999	0.9999	0.9999	0.9999	0.9999	1.0000	0.9999	0.9999	0.9999	1.0000	1.0000		1.3501
Yersinia pestis KIM ch	0.9997	1.0000	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998	0.9998	0.9997	0.9997	0.9996	0.9996	0.9995	0.9995	0.9996	

Fig. 5. Correlations (lower triangles) and Euclidean distances (upper triangles) between 17 bacterial signatures (column 1) described in Fig. 2. They give rise to a contrast-based phylogenetic tree **T16 cDNA** that exactly reproduces, at the genus level, the corresponding expanded and generally accepted tree **16S rRNA** reported in Fig. 6.

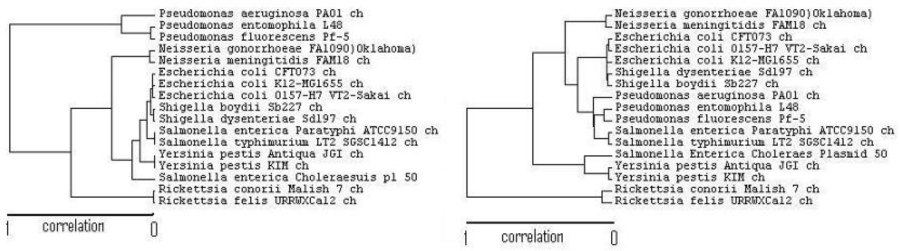

Pseudomonas aeruginosa PA01 ch
Pseudomonas entomophila L48
Pseudomonas fluorescens Pf-5
Neisseria gonorrhoeae FA1090)Oklahoma)
Neisseria meningitidis FAM18 ch
Escherichia coli CFT073 ch
Escherichia coli K12-MG1655 ch
Escherichia coli O157-H7 VT2-Sakai ch
Shigella boydii Sb227 ch
Shigella dysenteriae Sd197 ch
Salmonella enterica Paratyphi ATCC9150 ch
Salmonella typhimurium LT2 SGSC1412 ch
Yersinia pestis Antiqua JGI ch
Yersinia pestis KIM ch
Salmonella enterica Choleraesuis pl 50
Rickettsia conorii Malish 7 ch
Rickettsia felis URRWXCal2 ch
1 correlation 0
Neisseria gonorrhoeae FA1090)Oklahoma)
Neisseria meningitidis FAM18 ch
Escherichia coli CFT073 ch
Escherichia coli O157-H7 VT2-Sakai ch
Escherichia coli K12-MG1655 ch
Shigella dysenteriae Sd197 ch
Shigella boydii Sb227 ch
Pseudomonas aeruginosa PA01 ch
Pseudomonas entomophila L48
Pseudomonas fluorescens Pf-5
Salmonella enterica Paratyphi ATCC9150 ch
Salmonella typhimurium LT2 SGSC1412 ch
Salmonella Enterica Choleraes Plasmid 50
Yersinia pestis Antiqua JGI ch
Yersinia pestis KIM ch
Rickettsia conorii Malish 7 ch
Rickettsia felis URRWXCal2 ch
1 correlation 0

Fig. 6. 16S rRNA (left): Phylogenetic tree of seventeen (17) bacterial chromosomes shown in Table 2 generated by the CSRS method as before. The tree **T16 cDNA** (right) is identical to it at the genus level, except for minor difference at the species level that are still a matter of debate among biologists. (**cDNA** is used merely to indicate that the probes are complementary to fragments of the original genes.).

The trees generated by both CSRS and T16DNA were very much alike. Bacteria sharing the same genus name are grouped into a distinct clad separated from other genera at very high degree of resolution. It should be noted that, because of the problems of the above mentioned difficulties, the current rRNA-based method of phylogeny is not able to cluster bacterial phylogeny at this level of resolution. Additionally, polyphasic approaches that use multiple biomarkers are difficult to find for bacteria of diverse origins. A set of conserved genes in one group of bacteria, for example in the *Escherichia-Shigella-Salmonella* group, may be missing in other group, such as the highly degenerative *Rickettsia* group. When more and more bacteria genomes are being compared, the number of commonly shared conserved genes diminished accordingly. The lack of commonly shared conserved genes makes sequence comparison as a tool for phylogeny difficult. The CSRS phylogeny and the T16 DNA phylogeny are whole genome based. This feature bypasses the limitation of using commonly shared conserved genes, and thus makes direct comparisons of distant organisms possible. More interestingly, both methods predict the origin of the Salmonella plasmid 50 as being acquired horizontally, likely from some bacteria somewhat related to *Yesinia*. To the best of our knowledge, this is the first report ever showing the origin of this plasmid.

It must be emphasized that, although the two trees presented converge on support for a common conclusion, they are based on two totally different philosophical approaches. The CSRS-based tree is based on the biological assumption that the stop signal profiles would impose a restriction on genome expansion, whereas the 16T cDNA method is purely based on the physical chemistry (Gibbs energy) of DNA, with no biological assumptions being made. The 16T cDNA tree puts the *Pseudomonas* into the *Escherichia-Shigella-Salmonella* clad, which is the only inconsistent placement with the CSRS-tree and contradicts with the accepted concept of bacterial taxonomy. A plausible explanation of this fact is that a phylogenic tree is, like a real tree, in fact a 3D structure, while the current method of tree representation is only 2D. It is likely that the branch of *Pseudomonas* was forced to merge into the

Escherichia-Shigella-Salmonella clad when the data were collapsed into a 2D space (Wong et al. 2009). A better method to construct 3D phylogentic trees is being developed to explain this inconsistency.

4 Conclusions and Future Work

This paper continues the exploration of noncrosshybrizing sets developed for DNA Computing (Garzon et al., 2009; Garzon et al., 2009; Deaton et al., 2006) as a DNA-based indexing system for biological applications. We have shown that the recently developed technique of DNA indexing (Bobba et al, 2006; Garzon et al., 2005; Garzon et al., 2004b) can be successfully applied to infer *ab initio* phylogenetic trees that are solidly established and well accepted in biological phylogenesis for much larger families of bacteria. The major goal of this research is to develop a methodology for enabling arbitrary species classification based on so-called universal DNA chips that cover the entire spectrum of organisms, *known or unknown*. These DNA chips can be regarded as generalized barcodes. Although barcodes are, at best, expected to discriminate among species, they do not provide an insight into the more general and complex relationships among genomes captured by phylogentic relationships, or their complex intra- or inter-genomic relationships, let alone a basis for comparative genome-wide analysis. Our method allows in addition to species discrimination (which was the original purpose of the new contrast metric used here from (Garzon et al., 2009).)

The main result of this paper is strong evidence that digital genomic analyses are a scalable technique that can provide significant insights into pylogenetic questions. The new DNA-based technique for genomic identification offers several other advantages. First, it is much more effective in terms of cost and time than by traditional methods using traditional sophisticated bioinformatic analysis. For example, the 16S RNA trees are based on conserved genes, which make possible phylogenetic analysis only in the presence of highly conserved genes and therefore is unreliable or inapplicable to more distant organisms. By contrast, the DNA-chip method used herein is whole-genomic and thus applicable to arbitrary organisms. It has been further demonstrated that it may provide novel and principled insights into the phylogenesis of organisms hitherto inaccessible by current methods, such as a prediction of the origin of the *Salmonella plasmid 50* as being acquired horizontally, likely from some bacteria somewhat related to *Yesinia*. Second, it can be applied with newly available universal DNA chips readily available both *in vitro* by the PCR Selection protocol (Deaton et al., 2006; Chen et al., 2006) and *in silico* (Garzon et al., 2009; Garzon et al., 2009b; Garzon et al., 2004) through computer simulations in virtual test tubes (Garzon et al., 2004). This means, in particular, that they may well be able to provide a universal coordinate system to characterize very large groups (families, genera, and even phyla) of organisms on a common reference system, a veritable comprehensive “Atlas of Life”, as it is or as it could be on earth.

Acknowledgements

Many thanks to Abishek Logishetty and Jason Knisley for their help in producing the visualization of the signatures and trees above.

References

1. Adleman, L.: Molecular computation of solutions of combinatorial problems. Science 266, 1021–1024 (1994)
2. Blain, D., Garzon, M.H., Shin, S.Y., Zhang, B.T., Kashiwamura, S., Yamamoto, M., Kameda, A., Ohuchi, A.: Development, Evaluation and Benchmarking of Simulation Software for Biomolecule-based Computing. J. of Natural Computing 3(4), 427–442 (2004)
3. Bobba, K.C., Neel, A.J., Phan, V., Garzon, M.H.: "Reasoning" and "Talking" DNA: Can DNA understand english? In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 337–349. Springer, Heidelberg (2006)
4. Chen, J., Deaton, R., Garzon, M., Wood, D.H., Bi, H., Carpenter, D., Wang, Y.Z.: Characterization of Non-Crosshybridizing DNA Oligonucleotides Manufactured *in vitro*. J. of Natural Computing 5(2), 165–181 (2006)
5. Deaton, J., Chen, J., Garzon, M., Wood, D.H.: Test Tube Selection of Large Independent Sets of DNA Oligonucleotides R, pp. 152–166. World Publishing Co, Singapore (2006) (Volume dedicated to Ned Seeman on occasion of his 60th birthday)
6. Garzon, M.H., Wong, T.-Y., Phan, V.: DNA Chips for Species Identification and Biological Phylogenies. In: Deaton, R., Suyama, A. (eds.) DNA 15. LNCS, vol. 5877, pp. 55–66. Springer, Heidelberg (2009)
7. Garzon, M.H., Phan, V., Neel, A.: Optimal Codes for Computing and Self-Assembly. *Int.*. J. of Nanotechnology and Molecular Computing 1, 1–17 (2009b)
8. Garzon, M.H., Yan, H. (eds.): DNA 2007. LNCS, vol. 4848. Springer, Heidelberg (2008)
9. Garzon, M.H., Phan, V., Bobba, K.C., Kontham, R.: Sensitivity and capacity of microarray encodings. In: Carbone, A., Pierce, N.A. (eds.) DNA 11. LNCS, vol. 3892, pp. 81–95. Springer, Heidelberg (2006)
10. Garzon, M.H., Blain, D., Neel, A.J.: Virtual Test Tubes for Biomolecular Computing. J. of Natural Computing 3(4), 461–477 (2004)
11. Neel, A., Garzon, M.: Semantic Retrieval in DNA-Based Memories with Gibbs Energy Models. Biotechnology Progress 22(1), 86–90 (2006)
12. Ochman, H., Elwyn, S., et al.: Calibrating bacterial evolution. Proc. Natl. Acad. Sci. USA 96(22), 12638–12643 (1999)
13. Tulpan, D., Andronescu, M., Chang, S.B., Shortreed, M.R., Condon, A., Hoos, H.H., Smith, L.M.: Thermodynamically based DNA strand design, Nucleic Acids Res. Thermodynamically Based DNA Strand Design, Nucleic Acids Res. 33(15), 4951–4964 (2005)
14. Volff, J.N., Altenbuchner, J.: A new beginning with new ends: linearisation of circular chromosomes during bacterial evolution. FEMS Microbiol. Lett. 186(2), 143–150 (2000)
15. Woese, C., Fox, G.: Phylogenetic structure of the prokaryotic domain: the primary kingdoms. Proc. Natl. Acad. Sci. USA 74, 5088–5090 (1977)
16. Wong, T.Y., Fernandes, S., Sankhon, N., Leong, P.P., Kuo, J., Liu, J.K.: On the role of premature stop codons in bacterial evolution. J. Bacteriology (2009) (in press); Preliminary result presented at the 3rd Congress of FEMS (2008)
17. Zhou, F., Olman, V., Xu, Y.: Barcodes for Genomes and Applications. Bioinformatics 9, 546 (2008)

High-Fidelity DNA Hybridization Using Programmable Molecular DNA Devices

Nikhil Gopalkrishnan, Harish Chandran, and John Reif

Department of Computer Science, Duke University,
Durham, North Carolina 27708
{nikhil,harish,reif}@cs.duke.edu
http://www.cs.duke.edu

Abstract. The hybridization of complementary nucleic acid strands is the most basic of all reactions involving nucleic acids, but has a major limitation: the specificity of hybridization reactions depends critically on the lengths of the complementary pairs of strands and can drop to very low values for sufficiently long strands. This reduction in specificity occurs especially in the presence of *noise* in the form of other competing strands that have sequence segments identical to the target. This limits the scale and accuracy of biotechnology and nanotechnology applications which depend on hybridization reactions. Our paper develops techniques for ensuring specific high-fidelity DNA hybridization reactions for target strands of arbitrary length. Our protocol is executed autonomously, without external mediation and driven by a series of conversions of single stranded DNA into duplex DNA that help overcome kinetic energy traps, similar to DNA walkers.

Keywords: DNA hybridization, strand displacement.

1 Introduction

1.1 Motivation

The hybridization of complementary nucleic acid strands is the most basic of all reactions involving nucleic acids and a major component of most protocols involving nucleic acids. Indeed, hybridization reactions are the basis for much of biotechnology involving nucleic acids. For example, they are essential to many DNA enzymatic reactions such as restriction cuts, to PCR reactions used for amplification and for the operation of DNA hybridization arrays. Hybridization reactions are also the basis of DNA nanotechnology, which use hybridization of strands to form DNA tile nanostructures, as well as to bind DNA tile nanostructures together to form DNA lattices and to form DNA origami via hybridization between long scaffold strands and short staple strands.

However, the hybridization reaction has certain key limitations. The primary limitation is that the specificity of hybridization reactions (the likelihood that a given strand only hybridizes with its exact complementary strand) depends

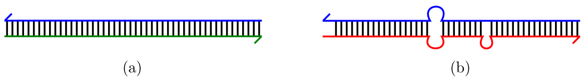


Fig. 1. Specificity limited by length: (a) Specific binding (b) Unspecific binding

critically on the lengths of the complementary pairs of strands. For long strands the difference in hybridization energies between a perfectly hybridized dsDNA and one with few mismatched bases is not significant. For example, in figure 1 the difference in stability for structures (a) and (b) is insignificant and at room temperature the relative concentrations of these structures are sensitive to relative concentrations of the single strands and hence binding fidelity of perfect complementary strands is low. While hybridization reactions in the appropriate solution conditions and temperature can have high-fidelity for moderate strand lengths (from 5 to 15 bases), the specificity of hybridization reactions can drop to very low values if the strands have sufficiently long length (say 25 or more bases). This reduction in specificity of hybridization reactions occurs especially in the presence of *noise* in the form of other competing strands that have sequence segments identical to the target. This limitation in the specificity of hybridization reactions depending on strand length significantly limits the scale and accuracy of biotechnology and nanotechnology applications which depend on hybridization reactions. For example, it limits the length of PCR primers, the length and thus the number of distinct strands in DNA hybridization arrays and the complexity of certain DNA nanostructures such as DNA origami.

1.2 Problem Statement: High-Fidelity DNA Hybridization

Let ssDNA denote a single stranded DNA and dsDNA denote a duplex DNA. Consider a solution containing distinct DNA sequences, with one of these sequences designated as a *target* s. We assume the particular known target DNA strand s is of relatively long length (say at least 60 to hundreds of bases). A target ssDNA in solution is said to be *completely hybridized* if all bases of the strand are (Watson-Crick) hybridized to corresponding complementary bases on other ssDNA, thus leaving no single stranded region on it. Note that multiple ssDNA may contribute complementary bases and thus cooperatively completely hybridize the target. The problem of *Exact High-Fidelity DNA Hybridization* is to completely hybridize each instance of s in solution while no instances of any other strand is completely hybridized. The problem of Exact High-Fidelity DNA Hybridization appears too stringent to be achievable in practice, since it does not allow for a small probability of failure or incomplete hybridizations nor does it allow for minor base mismatches. Hence we instead will take as our goal an approximate version of the High-Fidelity DNA Hybridization defined as follows.

The *Levenshtein distance* is a metric for measuring the edit difference between two sequences. In this paper the allowable edit operations on DNA sequences are insertion, deletion or substitution of a single base. A ssDNA in solution is b-*hybridized* if all but b bases of the strand are (Watson-Crick) hybridized to corresponding complementary bases on other ssDNA. Note that multiple ssDNA may contribute complementary bases and thus cooperatively b-hybridize the target. Given a fixed *success probability* p, and *base mismatch error* b (where $0 < p < 1$ and b is a fixed positive integer), the problem of (p, b)-*High-Fidelity DNA Hybridization* is to b-hybridize with probability at least p each instance of the target s in solution while no other strand is b-hybridized with probability greater than $1-p$ (Note: Typically, in our protocols, $1-p$ might be in the order of a few percentages and b might be a very small constant). Even this approximate version of High-Fidelity DNA Hybridization is quite challenging, as discussed in the first subsection.

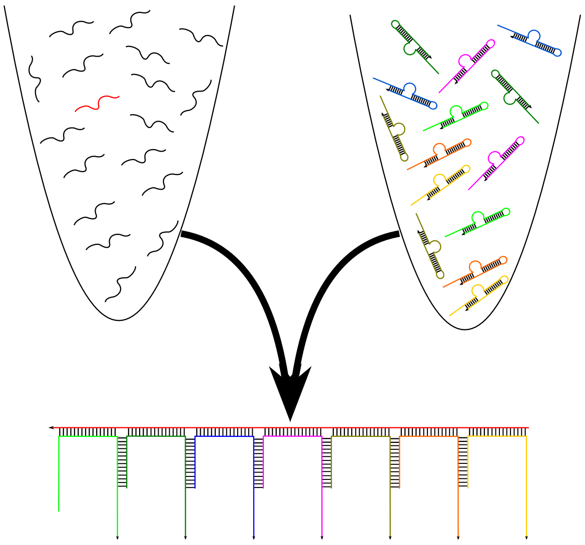


Fig. 2. High-fidelity hybridization problem

1.3 Our Results: Protocols for High-Fidelity DNA Hybridization Using DNA Devices

In this paper, we describe two protocols to achieve high-fidelity hybridization to an arbitrary given target DNA sequence. Our high-fidelity DNA hybridization protocols have the following favorable properties:

- Our protocols use only hybridization reactions of relatively short length (of approximately at most 15 bases), which are inherently highly specific.
- Our protocols are executed autonomously, without external mediation.

Our basic approach is to design DNA devices that essentially scan over strands in solution, subsegment by subsegment, and determine if one is indeed an instance of the given target strand s. This scanning is achieved by carefully designed relatively short sequences (called *checker sequences*) that hybridize to distinct contiguous subsequences on the target sequence (see fig. [2]). These checker strands perform successive *subsequence verification*. They are designed such that if the appropriate subsequence on one of them doesn't hybridize sufficiently to a specific subsegment of potential target strand, the subsequent *checker* strands do not hybridize to this potential target. Completion of the series of hybridizations indicates that the complete strand has been verified and hybridized.

To ensure protocols are executed autonomously, without external mediation, our high-fidelity DNA hybridization protocols are driven by a series of conversions of single stranded DNA into duplex DNA that help overcome kinetic energy traps, similar to DNA walkers (see [1],[2] and [3]). In addition to the design of our high-fidelity DNA hybridization protocol, we also discuss the kinetics of our hybridization reactions, reducing the overall kinetics to a series of well-understood strand-displacement reactions. Further, we describe a detailed design of an ongoing small-scale experimental demonstration of our protocols for high-fidelity hybridization. We also discuss potential applications for our protocols in molecular detection and DNA computing.

1.4 Notation

We facilitate a general symbolic design and description of our protocols and component DNA nanostructures by using the following notational conventions in text and figures:

- All DNA sequences will be represented by Latin letters and these letters may have subscripts, for example c_i.
- Subsequences are also denoted by letters and can also have subscripts, for eg. $a_i b_i = c_i$ where c_i is the concatenation of the subsequences a_i and b_i.
- The sequences are always written from $5'$ to $3'$. Arrows on DNA strands in the figures indicate $3'$ ends.
- Sequences denoted with the same letter that differ only in the subscript will be concatenations of subsequences that only differ in the subscript. For eg. $c_i = a_i d_{i-1}$ implies that the sequence c_{i+1} is a concatenation of a_{i+1} and d_i.
- A *bar* over a sequence indicates reverse complement of a sequence. For eg. $\bar{c}_i$ is the reverse complement of c_i and $\bar{b}_i \bar{a}_i$ is the reverse complement of $a_i b_i$.
- To decrease cluttering of figures, we only indicate the subsequences of one of the component strands of dsDNA.

2 First Protocol for High-Fidelity Hybridization

Let $s = r_n t_n r_{n-1} t_{n-1} \ldots r_i t_i \ldots r_1 t_1$ be a long DNA target strand, where r_i and t_i for $i = 1, \ldots, n$ are DNA subsequences. We wish to bind short checker sequences $c_i : i = 1, \ldots, n$ to s. Figure 3(a) indicates two consecutive checker sequences c_i above and c_{i+1} below. The expected secondary structure of these strands is also indicated in the figure. As an inductive hypothesis, assume that c_i is bound to the template strand s as shown in Figure 3(b). We claim that the appropriate complementary portion of strand c_{i+1}, $\bar{t}_{i+1} \bar{r}_{i+1}$ binds to $r_{i+1} t_{i+1}$ on s and the induction is advanced by one step.

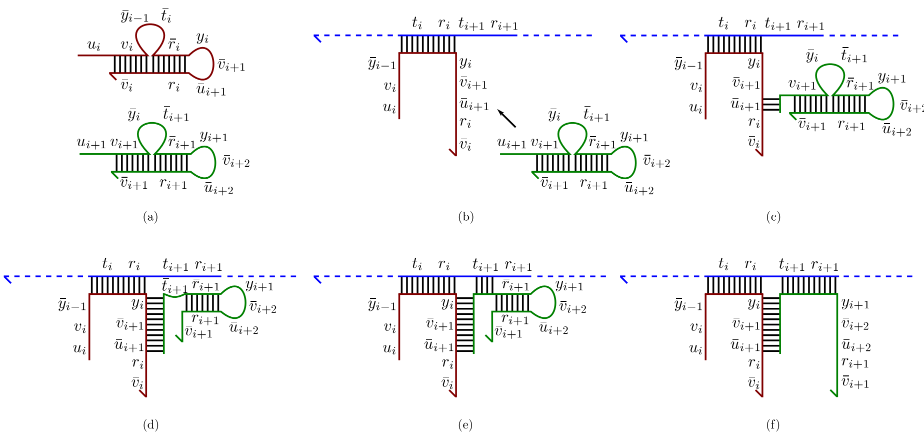


Fig. 3. First protocol: inductive step

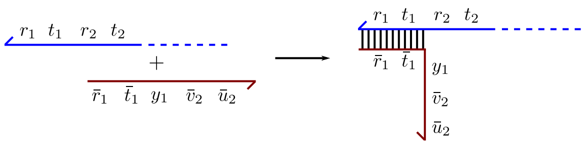


Fig. 4. First protocol: initiation step

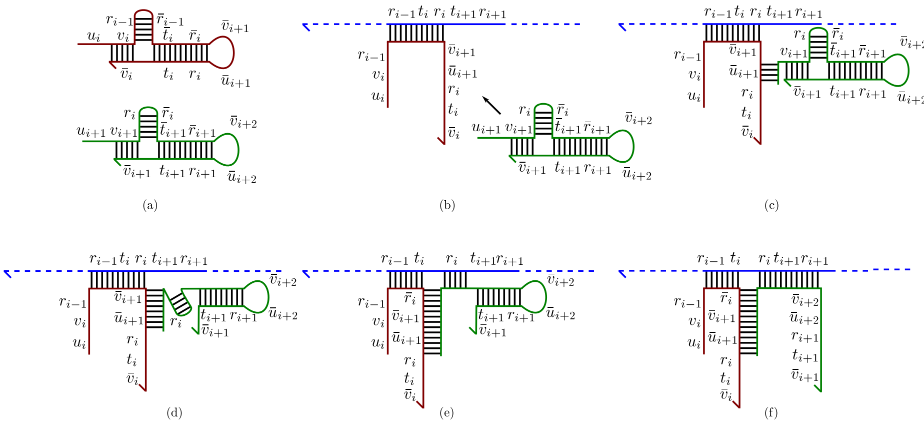
First, u_{i+1} on the checker strand c_{i+1} binds to its complementary region $\bar{u}_{i+1}$ which is a part of the c_i checker strand (Figure 3(c)). Through this toehold, a strand displacement reaction occurs, breaking the bond between v_{i+1} and $\bar{v}_{i+1}$. v_{i+1} is now attached to its complementary region $\bar{v}_{i+1}$ on the c_i checker strand. This opens the hairpin $\bar{y}_i\bar{t}_{i+1}$ allowing $\bar{y}_i$ to attach to y_i on the c_i checker strand (Figure 3(d)) and also $\bar{t}_{i+1}$ to attach to t_{i+1} on the template strand (Figure 3(e)). Now, another strand displacement reaction occurs via the toehold t_{i+1} on the template strand which breaks the bonds between r_{i+1} and $\bar{r}_{i+1}$ allowing $\bar{r}_{i+1}$ on the c_{i+1} checker strand to bind to r_{i+1} on the template strand (Figure 3(f)). This opens the hairpin $y_{i+1}\bar{v}_{i+2}\bar{u}_{i+2}$. Thus, the c_{i+1} checker strand is in the same conformation as the c_i checker strand was at the beginning of the induction step. The sequence $\bar{v}_{i+2}\bar{u}_{i+2}$ on the c_{i+1} checker strand can now open up the first hairpin on strand c_{i+2}, thus activating it and the process continues till all c_i are bound to the target s. If the potential target s does not have the appropriate sequence, some checker strand will not bind and hence the sequence of attachments will be halted.

The protocol is initiated when the initiator checker sequence $\bar{t}_1\bar{r}_1y_1\bar{v}_2\bar{u}_2$ is added to the solution and binds to r_1t_1 on the target strand (Figure 4). A potential source of error for the protocol is if the binding between v_i and $\bar{v}_i$ on checker sequence c_i dehybridizes due to thermal breathing causing c_i to bind to the potential target via the now exposed toehold $\bar{t}_i$. This might result in false positives, i.e. strands other than the target s might be falsely identified as the target. We overcome this potential pitfall in the next protocol where none of the subsequences on checker strands that are complementary to the target exist as single strands. They each exist as part of dsDNA. The tradeoff is the expected time to completion of the protocol, the second protocol is expected to be slower due to presence of three-way branch migrations as opposed to conventional toehold-mediated stand displacement.

3 Second Protocol for High-Fidelity Hybridization

As before, let $s = r_n t_n r_{n-1} t_{n-1} \ldots r_{i+1} t_{i+1} r_i t_i \ldots r_1 t_1$ be a long DNA target strand and $c_i : i = 1, \ldots, n$ be short checker sequences that bind to s. Figure 5(a) indicates two consecutive checker sequences c_i and c_{i+1} and their expected dominant secondary structure. As an inductive hypothesis, assume that c_i is bound to the template strand s as shown in Figure 5(b). We claim that the appropriate complementary portion of strand c_{i+1}, $\bar{r}_i \bar{t}_{i+1} \bar{r}_{i+1}$ binds to $r_{i+1} t_{i+1} r_i$ on s and the induction is advanced by one step. The chief difference between this protocol and the earlier one is that the incoming checker strand first strand invades from $5'$ to $3'$ on the template strand and then uses this toehold to strand invade the rest of the subsequence from $3\prime$ to $5'$. Note that this is not a simple toehold-mediated strand displacement, rather it is a three-way branch migration.

First, u_{i+1} on the c_{i+1} checker strand binds to its complementary region $\bar{u}_{i+1}$ on the c_i checker strand (Fig. 5(c)). Through this toehold, a strand displacement reaction occurs, breaking the bond between v_{i+1} and $\bar{v}_{i+1}$. v_{i+1} is now attached to its complementary region $\bar{v}_{i+1}$ on the c_i checker strand (Figure 5(d)). Now the hairpin structure r_i and $\bar{r}_i$ on the c_{i+1} checker strand can invade from $5'$ to $3'$ on the template strand s while at the same time attaching to $\bar{r}_i$ on the c_i checker strand (Fig. 5(e)). This breaks the bond between r_i on the template strand and $\bar{r}_i$ on the c_i checker strand. Now, a strand displacement reaction occurs via the toehold r_i on the template strand which opens a hairpin structure by breaking



r_{i-1} $\bar{r}_{i-1}$ u_i v_i $\bar{t}_i$ $\bar{r}_i$ $\bar{v}_{i+1}$ $\bar{v}_i$ t_i r_i $\bar{u}_{i+1}$
r_i $\bar{r}_i$ u_{i+1} v_{i+1} $\bar{t}_{i+1}$ $\bar{r}_{i+1}$ $\bar{v}_{i+2}$ $\bar{v}_{i+1}$ t_{i+1} r_{i+1} $\bar{u}_{i+2}$
(a)
$r_{i-1} t_i r_i t_{i+1} r_{i+1}$ r_{i-1} v_i u_i $\bar{v}_{i+1}$ $\bar{u}_{i+1}$ r_i t_i $\bar{v}_i$
(b)
(c)
(d)
(e)
(f)

Fig. 5. Second protocol: inductive step

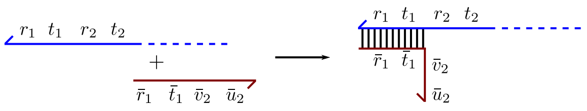

r_1 t_1 r_2 t_2
+
$\bar{r}_1$ $\bar{t}_1$ $\bar{v}_2$ $\bar{u}_2$
r_1 t_1 r_2 t_2
$\bar{r}_1$ $\bar{t}_1$ $\bar{v}_2$
$\bar{u}_2$

Fig. 6. Second protocol: initiation step

the bonds between $r_{i+1}t_{i+1}$ and $\bar{r}_{i+1}\bar{t}_{i+1}$ on the c_{i+1} checker strand, allowing $\bar{r}_{i+1}\bar{t}_{i+1}$ on the c_{i+1} checker strand to bind to $r_{i+1}t_{i+1}$ on the template strand (Fig. 5(f)). This opens the hairpin $\bar{v}_{i+2}\bar{u}_{i+2}$. Thus, the c_{i+1} checker strand is in the same conformation as the c_i checker strand was at the beginning of the induction step. The sequence $\bar{v}_{i+2}\bar{u}_{i+2}$ on the c_{i+1} checker strand can now open up the hairpin on checker strand c_{i+2}, thus activating it and the process continues till all c_i are bound to the template s. If the potential target s does not have the appropriate sequence, some checker strand will not bind and hence the sequence of attachments will be halted. The protocol is initiated when the initiator checker sequence $\bar{t}_1\bar{r}_1\bar{v}_2\bar{u}_2$ is added to the solution and binds to r_1t_1 on the template strand (Figure 6).

4 Potential Applications of High-Fidelity DNA Hybridization

Our set of checker sequences can be thought of as a high-fidelity rationally programmed aptamer for a specific DNA target sequence. Our checker sequences can be extended into functional nanostructures that interact with the target sequence, for example as a molecular cage that encapsulates the target molecule. The completion of subsequence verification can trigger other reactions and prove useful in molecular detection.

There are many significant applications of our protocols for High-Fidelity DNA Hybridization. These include significantly increasing the specificity of:

– DNA enzymatic reactions such as restriction cuts
– Priming of PCR reactions used for amplification
– DNA hybridization arrays
– Binding of the pads of DNA tile nanostructures together to form DNA lattices
– Hybridization between long scaffold strands and short sticker strands
– DNA computation reactions involving DNA hybridizations (see section 4.1)

4.1 Simulation of Deterministic Finite Automata

$M = \{\Sigma, S, s, F, \delta\}$ is a deterministic finite automaton (DFA) where Σ is the finite *input* alphabet, S is a finite set of *states*, $s \in S$ is the *start* state, $F \subseteq S$ is the set of *accepting* states and $\delta : S \times \Sigma \rightarrow S$ is the *transition* function. The *accepting* function of M, $\Delta_M : S \times \Sigma^* \rightarrow S$, is defined recursively for any

$a \in S, \beta \in \Sigma, x \in \Sigma^*$ as $\Delta_M(a, \beta x) = \Delta_M(\delta(a, \beta), x)$ and $\Delta_M(a, \epsilon) = a$. M accepts the language $L_M = \{w \in \Sigma^* | \Delta_M(s, w) \in F\}$.

Our protocols for high-fidelity hybridization can be adapted to implement any DFA. For simplicity assume that the input alphabet is $\Sigma = \{0, 1\}$. Also, assume that the DFA does not have any self-loops, i.e. transitions on input symbols that returns the DFA to the same state. It is a simple matter to convert any DFA with self-loops into an equivalent one with no self-loops by splitting states with self-loops into two states. We will adapt the protocol outlined in section 2 for our simulation. Let $w = \beta_1\beta_2 \ldots \beta_n$ be the input to the automaton, where $\beta_i \in \{0, 1\}$. We encode this input into the target strand along with an initiation sequence $\triangleright$ and a completion sequence $\triangleleft$ as $\triangleleft, \beta_n, \beta_{n-1}, \ldots, \beta_1, \triangleright$. In the notation used in section 2 the subsequence $r_i t_i$ encodes for the symbol β_i for $i = 1, 2, \ldots, n$. The protocol must check whether this input is in the language accepted by the automaton. The checker sequences will encode the transition function δ by associating a (state, symbol) combination with the appropriate next state. For example suppose the input sequence $\beta_1\beta_2 \ldots \beta_i$ has been consumed by the automaton and the current state is a. This current state will be encoded inside the i^{th} checker sequence as the subsequence $y_i\bar{v}_{i+1}\bar{u}_{i+1}$. The next transition $\delta(a, \beta_{i+1}) = b$ is executed by the $(i+1)^{\text{th}}$ checker sequence which contains the subsequences $u_{i+1}v_{i+1}\bar{y}_i$ (which codes for a), $\bar{t}_{i+1}\bar{r}_{i+1}$ (which codes for β_{i+1}) and $y_{i+1}\bar{v}_{i+2}\bar{u}_{i+2}$ (which codes for b) by hybridizing to complementary regions on the i^{th} checker sequence and the target. If the checker sequences successfully attach all the way to $r_n t_n$ this indicates that the target sequence encodes an input that should be accepted by the automaton. The special sequence $\triangleleft$ is initially in the form of a hairpin and the attachment of the last checker sequence opens the hairpin, which can be detected by fluoroscent emission due to the spatial decoupling of a fluorophore-quencher modified pair of DNA bases on the subsequence $\triangleleft$.

Note that instead of the correct $(i+1)^{\text{th}}$ checker sequence encoding the pair (a, β_{i+1}) an incorrect checker sequence that encodes $(a, \bar{\beta}_{i+1})$ may also attach to the previous checker sequence. However, it will not attach to the target and hence its second hairpin will remain intact, blocking further attachments. Thus, there is at least one of two choices of attachment at each step that further the process towards completion. If we assume equal probabilities of attachment for such competing checker sequences, an n length input has probability at least 2^{-n} of successful completion. This is a very low probability for long input sequences and alternate strategies to undo incorrect checker sequence attachment must be considered in this case. A key advantage of this simulation is that no matter what the input the same fixed set of checker sequences can check for acceptance of the input by the automaton. The number of such checker sequences is twice the number of states of the automaton. In fact, due to the fidelity of the hybridizations, multiple inputs may be processed in a parallel manner. By using fluorophores with distinct emission wavelengths on different targets we can detect multiple outputs in parallel. The simulation process is autonomous and does not use enzymes. It is easy to adapt this protocol to simulate non-deterministic finite automata and we leave the details to the reader.

5 Theoretical Analysis of Protocol Kinetics

The overall kinetics of our protocols can be reduced to the kinetics of a few typical hybridization and strand displacement reactions. This reduction is illustrated in figure 7, with corresponding forward and backward reaction rates. A key assumption is that sequences sequestered inside hairpin loops cannot participate in hybridization reactions. This idea was first used successfully in [4]. The simplest of the typical reactions is hybridization between two complementary single stranded regions in solution (Figure 7:(i),(v)) characterized by fast forward rates and slow backward rates for hybridizations of length ≥ 10. We also have *localized* hybridization between single stranded regions constrained

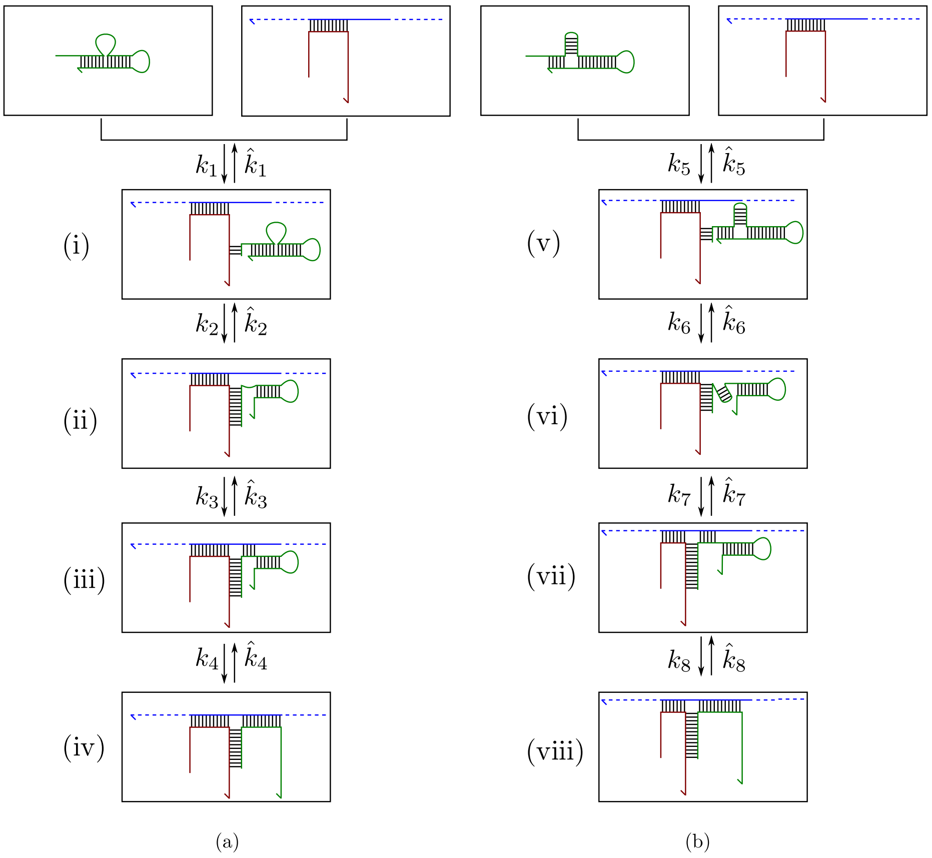


Fig. 7. Protocol kinetics partitioned into typical hybridization and strand displacement reactions

to lie close to each other (Figure 7(iii)), which has a significantly faster forward rate than unconstrained hybridization in solution. Then we have the well-understood toehold-mediated strand displacement (Figure 7:(ii),(iv),(vi),(viii)) with the slight caveat that the displaced sequence is constrained to be localized rather than floating away into the solution. This reaction is characterized by fast forward rates and slow backward rates, conditioned on appropriate length and sequence of the toehold. Finally we have a branch migration (Figure 7(vii)) whose forward and backward rates are not well-understood. However, we hypothesize that they are of comparable order. Note that the structure in figure 7(viii) is energetically favorable and will shift the balance of the structure from figure 7(vii) to 7(viii), driving our protocol forward. A more fine-grained analysis of the kinetics of our protocols requires kinetic modelling of these typical reactions.

Hybridization fueled DNA strand displacement reactions have been used in a variety of nanoscale protocols to achieve molecular transportation ([1], [5]), motors ([6]), detection ([4]) and computing ([7,8]). A thorough understanding of this fundamental process is key to the design of more complicated and involved dynamic DNA devices. Simple toehold-mediated strand displacement processes have been studied theoretically as one-dimensional random walks paving the way for simple stochastic models. A useful approach to understanding strand displacement is via kinetic computer simulations. Computer simulations can be a high-level tool for a cheap, quick and controlled investigation of the processes underlying strand displacement. Some of the information processing circuits demonstrated in [7] that use simple strand displacement processes have been simulated ([9]), but more general reactions involving multiple strands and strand exchange have not yet been studied. Involved DNA devices using branch migration reactions like the ones proposed in this paper for high-fidelity hybridization are a challenging and appropriate test case for such kinetic computer simulation studies. Experimental data about the kinetics of strand displacement is available ([10,5]) and can be used to create simulations that test the feasibility of the protocols proposed in this paper and improve their efficiency. These simulations can be thought of as a high-level testing and debugging environment that allow for improved designs of these involved protocols.

6 Conclusion

6.1 Experimental Verification

We propose a simple experiment using just two checker sequences (plus one initiator) to test if it can hybridize a target sequence with high-fidelity from an ensemble of moderately long sequences. We propose to use gel electrophoresis data and Förster resonance energy transfer (FRET) to test for the fully-complemented target sequence. Let the sequence of the target strand be $s = r_2 t_2 r_1 t_1 \triangleright$ with each of the subsequences relatively short. $\triangleright$ is a special subsequence to initiate the process. We can introduce *noise* in the form of other strands with sequences differing from that of the target. The two checker sequences are c_1 and c_2, like in figure 3(a) for the first protocol or like in figure 5(a) for the second protocol. There is an initiation sequence c_0 that binds to $\triangleright$ on the target and initiates binding of c_1. The terminal base at the $5'$ end of s can be modified with an emmitive fluorophore while the corresponding complementary base on the subsequence $\bar{r}_2$, part of c_2, can be modified with a corresponding quencher. If c_2 binds to the target s, the fluorophore-quencher pair will be brought in close proximity ($< 4nm$) and the fluoroscent emission of the fluorophore will be quenched signalling successful complete hybridization of the target. Appropriate controls can be devised for testing the specificity of binding of the checker sequences by modifying with an emmitive fluorophore the $5'$ ends of the *noisy* strands. In the absence of the target strand in the ensemble, no quenching of the fluoroscent signal should be observed.

6.2 Discussion

The problem of high-fidelity hybridization in long nucleic acid strands is a fundamental challenge with applications to a wide range of issues that arise frequently in biological nanotechnology. In this work, we have proposed two protocols for achieving highly specific hybridization to a specific target strand at the exclusion of all other strands in solution. We chiefly rely on hybridization between short segments of complementary DNA (which are inherently highly specific), strand displacement reactions and energy released by conversion of ssDNA to dsDNA to overcome kinetic traps in the form of meta-stable hairpins. We proposed simple experiments to test out our protocols in the base case where there are only two checker sequences that each verify half of the target strand. We wish to test whether our protocols will scale for much longer target sequences requiring many more checker sequences and hence we propose a computer based prediction of these protocols via thorough simulations of strand displacement reactions and dynamic behaviour of kinetic energy traps in the form of meta-stable hairpins.

Acknowledgments. This work was supported by NSF EMT grants CCF-0829797 and CCF-0829798.

References

1. Sherman, W., Seeman, N.: A Precisely Controlled DNA Biped Walking Device. Nano Letters 4, 1203–1207 (2004)
2. Yin, P., Yan, H., Daniell, X., Turberfield, A., Reif, J.: A Unidirectional DNA Walker Moving Autonomously Along a Linear Track. Angewandte Chemie International Edition 116(37), 5014–5019 (2004)
3. Tian, Y., He, Y., Chen, Y., Yin, P., Mao, C.: A DNAzyme That Walks Processively and Autonomously along a One-Dimensional Track. Angewandte Chemie International Edition 44(28), 4355–4358 (2005)
4. Dirks, R., Pierce, N.: Triggered Amplification by Hybridization Chain Reaction. Proceedings of the National Academy of Sciences of the United States of America 101(43), 15275–15278 (2004)

5. Turberfield, A., Mitchell, J., Yurke, B., Mills, A., Blakey, M., Simmel, F.: DNA Fuel for Free-Running Nanomachines. Physical Review Letters 90(11) (2003)
6. Yurke, B., Turberfield, A., Mills, A., Simmel, F., Neumann, J.: A DNA-fuelled Molecular Machine Made of DNA. Nature 406(6796), 605–608 (2000)
7. Zhang, D., Turberfield, A., Yurke, B., Winfree, E.: Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA. Science 318, 1121–1125 (2007)
8. Yin, P., Sahu, S., Turberfield, A.J., Reif, J.H.: Design of autonomous DNA cellular automata. In: Carbone, A., Pierce, N.A. (eds.) DNA 11. LNCS, vol. 3892, pp. 399–416. Springer, Heidelberg (2006)
9. Phillips, A., Cardelli, L.: A Programming Language for Composable DNA Circuits. Journal of The Royal Society Interface 6(11), 419–436 (2009)
10. Green, C., Tibbetts, C.: Reassociation Rate Limited Displacement of DNA Strands by Branch Migration. Nucleic Acids Research 9(8), 1905–1918 (1981)

Synthesizing Minimal Tile Sets for Patterned DNA Self-assembly

Mika Göös and Pekka Orponen

Department of Information and Computer Science
Aalto University School of Science and Technology (TKK)
P.O. Box 15400, FI-00076 Aalto, Finland
{mika.goos,pekka.orponen}@tkk.fi

Abstract. The Pattern self-Assembly Tile set Synthesis (PATS) problem is to determine a set of coloured tiles that self-assemble to implement a given rectangular colour pattern. We give an exhaustive branch-and-bound algorithm to find tile sets of minimum cardinality for the PATS problem. Our algorithm makes use of a search tree in the lattice of partitions of the ambient rectangular grid, and an efficient bounding function to prune this search tree. Empirical data on the performance of the algorithm shows that it compares favourably to previously presented heuristic solutions to the problem.

1 Introduction

An appealing methodology for bottom-up manufacturing of nanoscale structures and devices is to use a self-assembling system of DNA tiles [10] to build a scaffold structure on which functional units are deposited [4; 8; 13]. A systematic approach to the design of self-assembling DNA scaffold structures was proposed and experimentally validated by Park et al. in [7]. However, as pointed out by Ma & Lombardi in [5], that design is wasteful of tile types, i.e. generally speaking the same scaffold structures can be assembled also from fewer types of specially-manufactured DNA complexes, thus reducing the requisite laboratory work.

Ma & Lombardi [5] formulated the task of minimizing the number of DNA tile types required to implement a given 2-D pattern abstractly as a combinatorial optimization problem, the **Patterned self-Assembly Tile set Synthesis** (PATS) problem, and proposed two greedy heuristics for solving it. In this paper, we present a systematic branch-and-bound approach to exploring the space of feasible PATS tilings, and assess its computational performance also experimentally. The method compares favourably to the heuristics proposed by Ma & Lombardi, finding noticeably smaller or even provably minimal tile sets in a reasonable amount of computation time. However, as the experimental results in Section 4 show, the computational problem still remains quite challenging for large patterns.

Our considerations take place in the **abstract Tile Assembly Model** (aTAM) of Winfree and Rothemund [9; 11; 12] (Sect. 2.1). In the PATS problem [5] (Sect. 2.2), one associates a **colour** with each tile type and targets a specific coloured **pattern** within a rectangular assembly. The question is: given the desired colour pattern, what is the smallest set of (coloured) tile types that will self-assemble to implement it?

Our definition of the PATS problem restricts the self-assembly process to proceed in a uniform way. This simplification allows us to design efficient strategies for an exhaustive search (Sect. 3). For a pattern of size $m \times n$, we reduce the problem of finding a minimal tile set to the problem of finding a minimum-size **constructible** partition of $[m] \times [n]$. Here, constructibility of a partition can be verified in time polynomial in m and n. This leads us to construct a search tree in the lattice of partitions of the set $[m] \times [n]$ and to find pruning strategies for this search tree.

2 Preliminaries

2.1 The Abstract Tile Assembly Model

Our notation is derived from those of [1; 3; 11]. First, to simplify our notations, let $\mathcal{D} = \{N, E, S, W\}$ be the set of four functions $\mathbb{Z}^2 \to \mathbb{Z}^2$ corresponding to the cardinal directions (north, east, south, west) so that $N(x,y) = (x, y+1)$, $E(x,y) = (x+1, y)$, $S = N^{-1}$ and $W = E^{-1}$.

Let Σ be a set of **glue types** and $s : \Sigma \times \Sigma \to \mathbb{N}$ a **glue strength** function such that $s(\sigma_1, \sigma_2) = s(\sigma_2, \sigma_1)$ for all $\sigma_1, \sigma_2 \in \Sigma$. In this paper, we only consider glue strength functions for which $s(\sigma_1, \sigma_2) = 0$ if $\sigma_1 \neq \sigma_2$. A **tile type** $t \in \Sigma^4$ is a quadruple $(\sigma_N(t), \sigma_E(t), \sigma_S(t), \sigma_W(t))$ of glue types for each side of a unit square. Given a set Σ of glues, an **assembly** $\mathcal{A}$ is a partial mapping from $\mathbb{Z}^2$ to Σ^4. A **tile assembly system** (TAS) $\mathscr{T} = (T, \mathcal{S}, s, \tau)$ consists of a finite set T of tile types, an assembly $\mathcal{S}$ called the **seed assembly**, a glue strength function s and a **temperature** $\tau \in \mathbb{Z}^+$ (we use $\tau = 2$).

To formalize the self-assembly process, we first fix a TAS $\mathscr{T} = (T, \mathcal{S}, s, \tau)$. For two assemblies $\mathcal{A}$ and $\mathcal{A}'$ we write $\mathcal{A} \to_{\mathscr{T}} \mathcal{A}'$ if there exists a pair $(x,y) \in \mathbb{Z}^2$ and a tile $t \in T$ such that $\mathcal{A}' = \mathcal{A} \cup \{((x,y),t)\}$, where the union is disjoint, and

$$\sum_D s(\sigma_D(t), \sigma_{D^{-1}}(\mathcal{A}(D(x,y)))) \;\geq\; \tau \;, \tag{1}$$

where D ranges over those directions in $\mathcal{D}$ for which $\mathcal{A}(D(x,y))$ is defined. This is to say that a new tile can be adjoined to an assembly $\mathcal{A}$ if the new tile shares a common boundary with tiles that bind it into place with total strength at least τ.

Let $\to^*_{\mathscr{T}}$ be the reflexive transitive closure of $\to_{\mathscr{T}}$. A TAS $\mathscr{T}$ **produces** an assembly $\mathcal{A}$ if $\mathcal{A}$ is an **extension** of the seed assembly $\mathcal{S}$, that is if $\mathcal{S} \to^*_{\mathscr{T}} \mathcal{A}$. Let us denote by $\operatorname{Prod} \mathscr{T}$ the set of all assemblies produced by $\mathscr{T}$. This way, the pair $(\operatorname{Prod} \mathscr{T}, \to^*_{\mathscr{T}})$ forms a partially ordered set. We say that a TAS $\mathscr{T}$ is **deterministic** if for any assembly $\mathcal{A} \in \operatorname{Prod} \mathscr{T}$ and for every $(x,y) \in \mathbb{Z}^2$ there exists at most one $t \in T$ such that $\mathcal{A}$ can be extended with t at position (x,y). A TAS $\mathscr{T}$ is deterministic precisely when $\operatorname{Prod} \mathscr{T}$ is a lattice. Also, the maximal elements in $\operatorname{Prod} \mathscr{T}$ are such assemblies $\mathcal{A}$ that can not be further extended, that is, there do not exist assemblies $\mathcal{A}'$ such that $\mathcal{A} \to_{\mathscr{T}} \mathcal{A}'$. These maximal elements are called **terminal assemblies**. We denote by $\operatorname{Term} \mathscr{T}$ the set of terminal assemblies of $\mathscr{T}$. If all **assembly sequences**

$$\mathcal{S} \to_{\mathscr{T}} \mathcal{A}_1 \to_{\mathscr{T}} \mathcal{A}_2 \to_{\mathscr{T}} \cdots \tag{2}$$

terminate and $\operatorname{Term} \mathscr{T} = \{\mathcal{P}\}$ for some assembly $\mathcal{P}$, we say that $\mathscr{T}$ **uniquely produces** $\mathcal{P}$.

2.2 The PATS Problem

In this paper we restrict our attention to designing minimal tile assembly systems that construct a given pattern in a finite rectangular m by n grid $[m] \times [n] \subseteq \mathbb{Z}^2$. This problem was first discussed by Ma & Lombardi [5].

Definition 1 (Pattern self-Assembly Tile set Synthesis (PATS) [5]).

Given: *A k-colouring $c : [m] \times [n] \to [k]$.*
Find: *A tile assembly system $\mathscr{T} = (T, \mathcal{S}, s, 2)$ such that*

P1. *The tiles in T have bonding strength 1.*
P2. *The domain of $\mathcal{S}$ is $[0,m] \times \{0\} \cup \{0\} \times [0,n]$ and all the terminal assemblies have the domain $[0,m] \times [0,n]$.*
P3. *There exists a colouring $d : T \to [k]$ such that for each terminal assembly $\mathcal{A} \in \operatorname{Term} \mathscr{T}$ we have $d(\mathcal{A}(x,y)) = c(x,y)$ for all $(x,y) \in [m] \times [n]$.*

In particular, we are interested in the **minimal solutions** (in terms of $|T|$) to the PATS problem. By the same token, we can make the following assumption:

Assumption 1. In our TASs, every tile participates in assembling some terminal assembly.

Ma & Lombardi show a certain derivative of the above optimization problem NP-hard in [6]. However, to our knowledge, a proof of the NP-hardness of the PATS problem as stated above is lacking.

As an illustration, we construct a part of the Sierpinski triangle with a 4-tile TAS in Figure 1. We use natural numbers as glue labels in our figures.

Due to constraint *P1* the self-assembly process proceeds in a uniform manner directed from south-west to north-east. This paves the way for a simple characterization of deterministic TASs in the context of the PATS problem.

Proposition 1. *Solutions $\mathscr{T} = (T, \mathcal{S}, s, 2)$ of the PATS problem are deterministic precisely when for each pair of glue types $(\sigma_1, \sigma_2) \in \Sigma^2$ there is at most one tile type $t \in T$ so that $\sigma_S(t) = \sigma_1$ and $\sigma_W(t) = \sigma_2$.*

The following simple observation reduces the work needed in finding minimal solutions of the PATS problem.

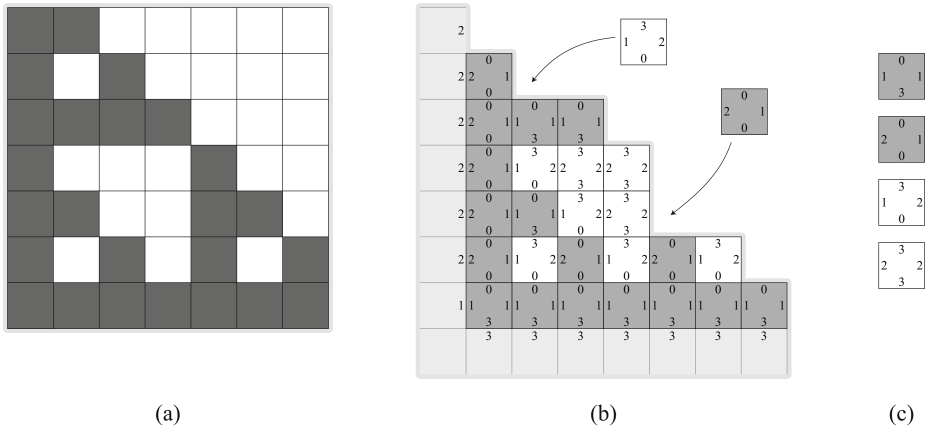


Fig. 1. (a) A finite subset of the discrete Sierpinski triangle. This 2-colouring of the set $[7] \times [7]$ defines an instance of the PATS problem. (b) Assembling the Sierpinski pattern with a TAS that has an appropriate seed assembly and a (coloured) tile set shown in (c).

Lemma 1. *The minimal solutions of the PATS problem are deterministic TASs.*

Proof. Suppose, for the sake of contradiction, that $\mathscr{N} = (T, \mathcal{S}, s, 2)$ is a minimal solution to a PATS problem and is not deterministic. By the above proposition let tiles $t_1, t_2 \in T$ be such that $\sigma_S(t_1) = \sigma_S(t_2)$ and $\sigma_W(t_1) = \sigma_W(t_2)$. One can now check that the simplified TAS $\mathscr{N}' = (T \setminus \{t_2\}, \mathcal{S}, s, 2)$ is a solution to the original PATS problem [2]. This violates the minimality of $|T|$.

Assumption 2. We consider only deterministic TASs in the sequel.

3 A Branch-and-Bound Algorithm

We describe an exact algorithm to find minimal solutions to the PATS problem. We extend the methods of [5] to obtain an exhaustive branch-and-bound (B&B) algorithm. The idea of Ma & Lombardi [5] (following experimental work of [7]) is to start with an **initial tile set** that consists of $m \cdot n$ different tiles, one for each of the grid positions in $[m] \times [n]$. Their algorithm then proceeds to merge tile types in order to minimize $|T|$. We formalize this search process as an exhaustive search in the set of all partitions of the set $[m] \times [n]$. In the following, we let a PATS instance be given by a fixed k-coloured pattern $c : [m] \times [n] \to [k]$.

3.1 The Search Space

Let X be the set of partitions of the set $[m] \times [n]$. For partitions $P, P' \in X$ we define a relation $\sqsubseteq$ so that

$$P \sqsubseteq P' \iff \forall p' \in P' : \ \exists p \in P : \ p' \subseteq p \ . \qquad (3)$$

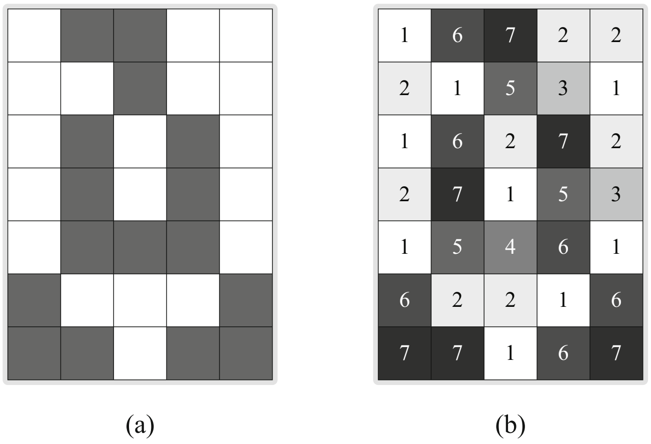


Fig. 2. (a) Partition A. (b) A partition M that is a refinement of A with $|M| = 7$ parts.

Now, $(X, \sqsubseteq)$ is a partially ordered set, and in fact, a lattice. If $P \sqsubseteq P'$ we say that P' is a **refinement** of P, or that P is **coarser** than P'. Note that $P \sqsubseteq P'$ implies $|P| \leq |P'|$.

The colouring c induces a partition $P(c) = \{c^{-1}(\{i\}) \mid i \in [k]\}$ of the set $[m] \times [n]$. In addition, since every (deterministic) solution $\mathscr{T} = (T, \mathcal{S}, s, 2)$ of the PATS problem uniquely produces some assembly $\mathcal{A}$, we associate with $\mathscr{T}$ a partition $P(\mathscr{T}) = \{\mathcal{A}^{-1}(\{t\}) \mid t \in \mathcal{A}([m] \times [n])\}$. Here, $|P(\mathscr{T})| = |T|$ due to our Assumptions 1 and 2. With this terminology, the condition *P3* in the definition of the PATS problem is equivalent to requiring that a TAS $\mathscr{T}$ satisfies

$$P(c) \sqsubseteq P(\mathscr{T}) \ . \tag{4}$$

We say that a partition $P \in X$ is **constructible** if $P = P(\mathscr{T})$ for some deterministic TAS $\mathscr{T}$ with properties *P1* and *P2*. With this, we can rephrase our goal from the point of view of using partitions as the fundamental search space.

Proposition 2. *A minimal solution to the PATS problem corresponds to a partition $P \in X$ such that P is constructible, $P(c) \sqsubseteq P$ and $|P|$ is minimal.*

For example, the 2-coloured pattern in Figure 2a defines a 2-part partition, A, say. The 7-part partition M in Figure 2b is a refinement of A ($A \sqsubseteq M$) and in fact, M is constructible (see Figure 3b) and corresponds to a minimal solution of the PATS problem defined by the pattern A.

3.2 Determining Constructibility

In this section we give an algorithm for deciding the constructibility of a given partition in polynomial time. To do this, we use the concept of most general (or least constraining) tile assignments. In the following, we write $f(p)_D$ instead of $\sigma_D(f(p))$.

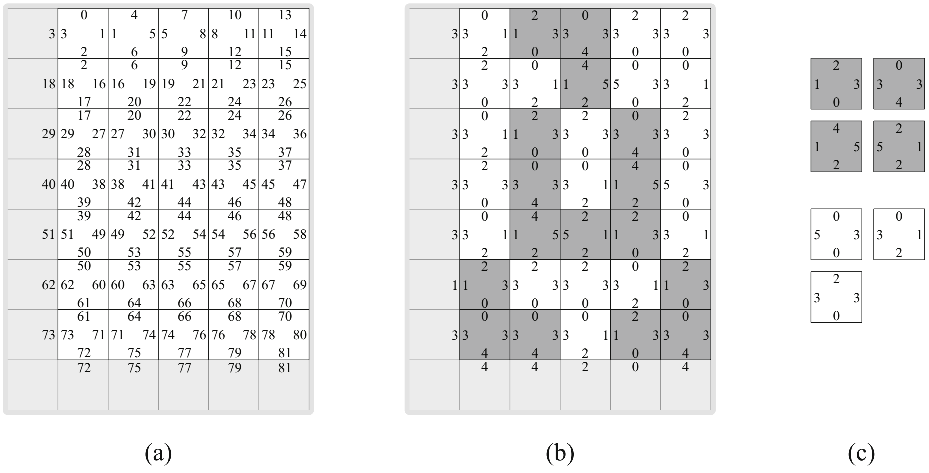


Fig. 3. (a) A MGTA for the constructible initial partition I (with a seed assembly in place). (b) Finished assembly for the pattern from Figure 2a. The tile set to construct this assembly is given in (c).

Definition 2. *Given a partition P of the set $[m] \times [n]$, a* ***most general tile assignment*** *(MGTA) is a function $f : P \to \Sigma^4$ such that*

A1. When every position in $[m] \times [n]$ is assigned a tile type according to f, any two adjacent positions agree on the glue type of the side between them.
A2. For all assignments $g : P \to \Sigma^4$ satisfying A1 we have

$$f(p_1)_{D_1} = f(p_2)_{D_2} \quad \Longrightarrow \quad g(p_1)_{D_1} = g(p_2)_{D_2} \tag{5}$$

for all $(p_1, D_1), (p_2, D_2) \in P \times \mathcal{D}$.

To demonstrate this concept we present a most general tile assignment $f : I \to \Sigma^4$ for the **initial partition** $I = \{\{a\} \mid a \in [m] \times [n]\}$ in Figure 3a and a MGTA for the partition of Figure 2b in Figure 3b.

Given a partition $P \in X$ and a function $f : P \to \Sigma^4$, we say that $g : P \to \Sigma^4$ is obtained from f by **merging glues** a **and** b if g coincides with f except that $g(p)_D = a$ if $f(p)_D = b$.

A most general tile assignment for a partition $P \in X$ can be found as follows. We start with a function $f_0 : P \to \Sigma^4$ that assigns to each tile edge a unique glue type, or in other words, a function f_0 so that the mapping $(p, D) \mapsto f_0(p)_D$ is injective. Next, we go through all pairs of adjacent positions in $[m] \times [n]$ (in some order) and require their matching sides to have the same glue type by merging the corresponding glues. This process generates a sequence of functions $f_0, f_1, f_2, \ldots, f_N = f$ and terminates after $N \leq 2mn$ steps.

Lemma 2. *The above algorithm generates a most general tile assignment.*

Corollary 1. *For a given partition, MGTAs are unique up to relabeling of the glue types.*

The detailed proofs of the above claims are given in [2].

For each partition P, we take **the MGTA for** P to be some canonical representative from the class of MGTAs for P.

We now give the (polynomial time decidable) conditions for a partition to be constructible in terms of MGTAs.

Lemma 3. *A partition $P \in X$ is constructible iff the MGTA $f : P \to \Sigma^4$ for P is injective and the tile set $f(P)$ is deterministic in the sense of Proposition 1.*

Proof. See [2] for the details.

3.3 An Initial Search DAG

Our algorithm performs an exhaustive search in the lattice $(X, \sqsubseteq)$ searching for constructible partitions. In the search, we maintain and incrementally update MGTAs for every partition we visit. First, we describe simple branching rules to obtain a rooted directed acyclic graph search structure and later give rules to prune this DAG to a node-disjoint search tree.

The root of the DAG is taken to be the initial partition I that is always constructible. For each partition $P \in X$ we next define the set $C(P) \subseteq X$ of children of P. Our algorithm always proceeds by combining parts of the partition currently being visited, so for each $P' \in C(P)$ we will have $P' \sqsubseteq P$. Say we visit a partition $P \in X$. We have two possibilities:

C1. P **is constructible:**
   1. If P is not a refinement of the target pattern $P(c)$, that is if $P(c) \not\sqsubseteq P$, we can drop this branch of the search, since no possible descendant $P' \sqsubseteq P$ can be a refinement of $P(c)$ either. (i.e. $C(P) = \varnothing$)
   2. In case $P(c) \sqsubseteq P$, we can use the MGTA for P to give a concrete solution to the PATS problem instance defined by the colouring c. To continue the search and to find more optimal solutions we consider each pair of parts $\{p_1, p_2\} \subseteq P$ in turn and recursively visit the partition $P[p_1, p_2]$ where the two parts are combined. In fact, by the above analysis, it is sufficient to consider only pairs of the same colour:

$$C(P) = \{P[p_1, p_2] \mid p_1, p_2 \in P,\ \ p_1 \neq p_2,\ \ \exists k \in P(c) : p_1, p_2 \subseteq k\} \ . \quad (6)$$

C2. P **is not constructible:** In this case the MGTA f for P gives $f(p_1)_S = f(p_2)_S$ and $f(p_1)_W = f(p_2)_W$ for some parts $p_1 \neq p_2$. We continue the search from partition $P[p_1, p_2]$:

$$C(P) = \{P[p_1, p_2]\} \ . \quad (7)$$

To guarantee that our algorithm finds the optimal solution in the case C2 above, we need the following [2].

Lemma 4. *Let $P \in X$ be a non-constructible partition, f the MGTA for P and $p_1, p_2 \in P$, $p_1 \neq p_2$, parts such that $f(p_1)_S = f(p_2)_S$ and $f(p_1)_W = f(p_2)_W$. For all constructible $C \sqsubseteq P$ we have $C \sqsubseteq P[p_1, p_2]$.*

3.4 Pruning the DAG to a Search Tree

Computational resources should be saved by not visiting any partition twice. To keep the branches in our search structure node-disjoint, we maintain a list of graphs that store restrictions on the choices the search can make.

For each partition $P \sqsupseteq P(c)$ we associate a family of undirected graphs $\{G_k^P\}_{k\in P(c)}$, one for each colour region of the pattern $P(c)$. Every part in P is represented by a vertex in the graph corresponding to the colour of the part. More formally, the vertex set $V(G_k^P)$ is taken to be those parts $p \in P$ for which $p \subseteq k$. (So now, $\bigcup_{k\in P(c)} V(G_k^P) = P$.) An edge $\{p_1, p_2\} \in E(G_k^P)$ indicates that the parts p_1 and p_2 are not allowed ever to be combined in the search branch in question. When we start our search with the initial partition I, the edge sets are initially empty, $E(G_k^I) = \varnothing$. At each partition P, the graphs $\{G_k^P\}_{k\in P(c)}$ have been determined inductively and the graphs for those children $P' \in C(P)$ that we visit are defined as follows.

D1. **If P is constructible:** We choose some ordering $\{p_i, q_i\}$, $i = 1, \ldots, N$, for similarly coloured pairs of parts. Define $l_i \in P(c)$, $1 \le i \le N$ to be the colour of the pair $\{p_i, q_i\}$, so that $p_i, q_i \subseteq l_i$. Now, we visit a partition $P[p_i, q_i]$ if and only if $\{p_i, q_i\} \notin E(G_{l_i}^P)$. When we decide to visit a child partition $P' = P[p_j, q_j]$, we define the edge sets $\{E(G_k^{P'})\}_{k\in P(c)}$ as follows:
   1. We start with the graphs $\{G_k^P\}_{k\in P(c)}$ and add the edges $\{p_i, q_i\}$ for all $1 \le i < j$ to their corresponding graphs. Call the resulting graphs $\{G_k^\star\}_{k\in P(c)}$.
   2. Finally, as we combine the parts p_j and q_j to obtain the partition $P[p_j, q_j]$, we merge the vertices p_j and q_j in the graph $G_{l_j}^\star$. The graphs $\{G_k^{P'}\}_{k\in P(c)}$ follow as a result.

D2. **If P is not constructible:** Here, the MGTA for P suggests a single child partition $P' = P[p_1, p_2]$ for some $p_1, p_2 \subseteq l \in P(c)$. If $\{p_1, p_2\} \in E(G_l^P)$, we terminate this branch of the search. Otherwise, we define the graphs $\{G_k^{P'}\}_{k\in P(c)}$ to be the graphs $\{G_k^P\}_{k\in P(c)}$, except that in $G_l^{P'}$ the vertices p_1 and p_2 have to be merged.

One can see that the outcome of this pruning process is a search tree that has node-disjoint branches and one in which every possible constructible partition is still guaranteed to be found. Figure 4 presents a sketch of the search tree.

Note that we are not usually interested in finding every constructible partition $P \in X$, but only in finding a minimal one (in terms of $|P|$). Next, we give an efficient method to lower-bound the partition sizes of a given search branch.

3.5 The Bounding Function

Given a root $P \in X$ of some subtree of the search tree, we ask: What is the smallest partition that can be found from this subtree? The nodes in the subtree rooted at P consists of those partitions $P' \sqsubseteq P$ that can be obtained from P by merging pairs of parts that are not forbidden by the graphs $\{G_k^P\}_{k\in P(c)}$.

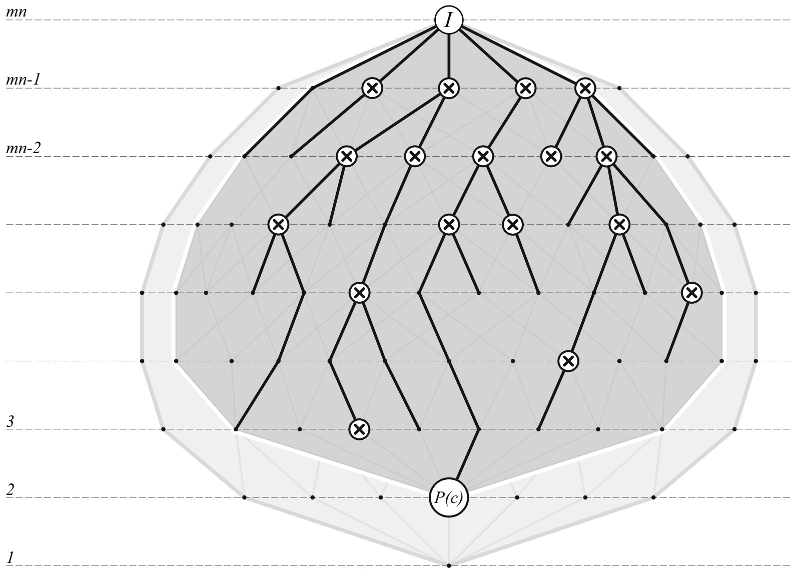


Fig. 4. The search tree in the lattice $(X, \sqsubseteq)$. We start with the initial partition I of size $|I| = mn$. The partition $P(c)$ defines the PATS problem instance: We search for constructible partitions (drawn as crosses) in the sublattice (shaded with darker grey) consisting of those partitions that are refinements of $P(c)$. The search tree branches only at the constructible partitions and the tree branches are node-disjoint.

This merging process halts precisely when all the graphs $\{G_k^{P'}\}_{k \in P(c)}$ have beed reduced into cliques. As is well known, the size of the smallest clique that a graph G can be turned into by merging non-adjacent vertices is given by the **chromatic number** $\chi(G)$ of the graph G. This immediately gives the following.

Proposition 3. *For every $P' \sqsubseteq P$ in the subtree rooted at P and constrained by $\{G_k^P\}_{k \in P(c)}$, we have*

$$\sum_{k \in P(c)} \chi(G_k^P) \leq |P'| . \tag{8}$$

Determining the chromatic number of an arbitrary graph is an NP-hard problem. Fortunately, we can restrict our graphs to be of a special form: graphs that consist only of a clique and some isolated vertices. For these graphs, the chromatic numbers are given by the sizes of the cliques.

To see how to maintain graphs in this form, consider as a base case the initial partition I. Here, $E(G_k^I) = \varnothing$ for all $k \in P(c)$, so G_k^I is of our special form—it has a clique of size 1. For a general partition P, we go through the branching rules D1-D2.

D1: P **is constructible:** Since we are allowed to choose an arbitrary ordering $\{p_i, q_i\}$, $i = 1, \ldots, N$, for the children $P[p_i, q_i]$, we design an ordering that preserves the special form of the graphs. For a graph G of our special form,

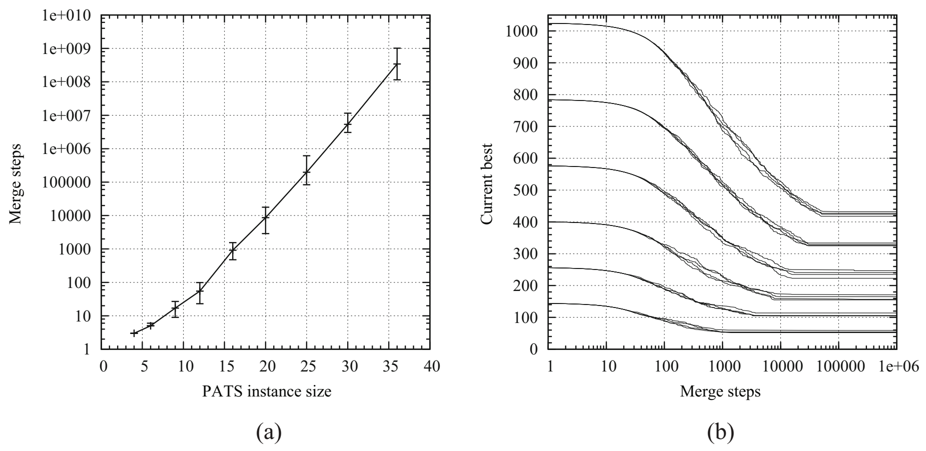


Fig. 5. (a) Running time of the algorithm (as measured by the number of merge operations) to solve random 2-coloured near-square-shaped instances of the PATS problem. (b) Evolution of the tile set size of the "current best solution" for several large random 2-coloured instances of the PATS problem.

let $K(G) \subseteq V(G)$ consist of those vertices that are part of the clique in G. In the algorithm, we first set $H_k = G_k^P$ for all $k \in P(c)$ and repeat the following process until every graph H_k is a complete clique.

1. Pick some colour $k \in P(c)$ and an isolated vertex $v \in V(H_k) \setminus K(H_k)$.
2. Process the pairs $\{v, u\}$ for all $u \in K(H_k)$ in some order. By the end, update H_k to include all the edges $\{v, u\}$ that were just processed (the size of the clique in H_k increases by one).

D2: P **is not constructible:** It is easy to see that if P is of our special form, so is $P' = P[p_1, p_2]$.

4 Results

The running time of our B&B algorithm is proportional—up to a polynomial factor—to the number of partitions the algorithm visits. Hence, we measure the running time in terms of the number of merge operations performed in the search. Figure 5a presents the running time of the algorithm to find a minimal solution for random 2-coloured instances of the PATS problem. The algorithm was executed for instance sizes $2 \times 2, 2 \times 3, 3 \times 3, \cdots, 5 \times 6$ and 6×6; the 20th and 80th percentiles are shown alongside the median of 21 separate runs for each instance size. For the limiting case 6×6, the algorithm spent on the order of two hours of (median) computing time on a 2,61 GHz AMD processor.

Even though B&B search is an exact method, it can be used to find approximate solutions by running it for a suitable length of time. Figure 5b illustrates how the best solution found up to a point develops as increasingly many steps of the algorithm are run. The figure provides data on random 2-coloured instances of sizes from 12×12 up to 32×32. Because we begin our search from the initial

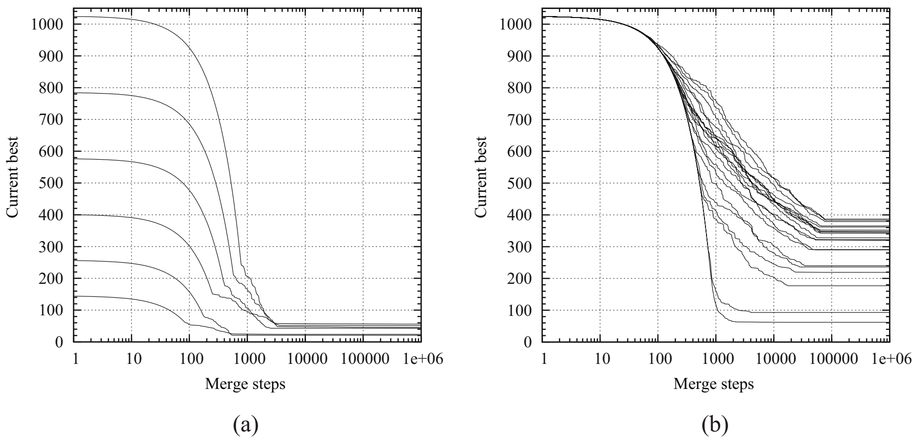


Fig. 6. Evolution of the "current best solution" for (a) the Sierpinski pattern and for (b) the binary counter pattern. Randomization in the DFS has a clear effect on the performance of the algorithm in the case of the binary counter pattern.

partition, the best solution at the first step is precisely equal to the instance size. For each size, several different patterns were used. The algorithm was cut off after 10^6 steps. By this time, an approximate reduction of 58% in the size of the tile set was achieved (cf. a reduction of 43.5% in [5]).

Next, we consider two well known examples of structured patterns: the discrete Sierpinski triangle (part of which was shown in Figure 1) and the binary counter (see Figure 1 in [11]). A tile set of size 4 is optimal for both of these patterns. First, for the Sierpinski pattern, we get a tile reduction of well over 90% (cf. 45% in [5]) in Figure 6a. We used the same cutoff threshold and instance sizes as in Figure 5b. Our description of the B&B algorithm leaves some room for randomization in deciding which search branch a DFS is to explore next. This randomization does not seem to affect the search dramatically when considering the Sierpinski pattern—the separate single runs in Figure 6a are representative of an average randomized run. By contrast, for the binary counter pattern, randomized runs for single instance size do make a difference. Figure 6b depicts several seperate runs for instance size 32×32. This suggests that, as is characteristic of DFS traversal, restarting the algorithm with a different random seed may help with large instances that have small optimal solutions.

5 Conclusion

We have presented an exact branch-and-bound algorithm for finding minimum-size tile sets that self-assemble a given k-coloured pattern in a uniform self-assembly setting. Simulation results indicate that our algorithm is able to find provably minimal tile sets for random instances of sizes up to 6×6 and can give approximate solutions for larger instances as well.

References

[1] Adleman, L., Cheng, Q., Goel, A., Huang, M.-D., Kempe, D., de Espanés, P.M., Rothemund, P.W.K.: Combinatorial optimization problems in self-assembly. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002), pp. 23–32. ACM, New York (2002)

[2] Göös, M., Orponen, P.: Synthesizing minimal tile sets for patterned DNA self-assembly. A detailed version, arXiv:0911.2924

[3] Lathrop, J.I., Lutz, J.H., Summers, S.M.: Strict self-assembly of discrete Sierpinski triangles. Theoretical Computer Science 410(4-5), 384–405 (2009)

[4] Lin, C., Liu, Y., Rinker, S., Yan, H.: DNA tile based self-assembly: building complex nanoarchitectures. Chem. Phys. Chem. 7(8), 1641–1647 (2006)

[5] Ma, X., Lombardi, F.: Synthesis of tile sets for DNA self-assembly. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(5), 963–967 (2008)

[6] Ma, X., Lombardi, F.: On the computational complexity of tile set synthesis for DNA self-assembly. IEEE Transactions on Circuits and Systems II: Express Briefs 56(1), 31–35 (2009)

[7] Park, S.H., Pistol, C., Ahn, S.J., Reif, J.H., Lebeck, A.R., Dwyer, C., LaBean, T.H.: Finite-size, fully addressable DNA tile lattices formed by hierarchical assembly procedures. Angewandte Chemie International Edition 45(5), 735–739 (2006)

[8] Park, S.H., Yan, H., Reif, J.H., LaBean, T.H., Finkelstein, G.: Electronic nanostructures templated on self-assembled DNA scaffolds. Nanotechnology 15, S525–S527 (2004)

[9] Rothemund, P.W.K.: Theory and Experiments in Algorithmic Self-assembly. PhD thesis, University of Southern California (2001)

[10] Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature 440, 297–302 (2006)

[11] Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000), pp. 459–468. ACM, New York (2000)

[12] Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology (1998)

[13] Yan, H., Park, S.H., Finkelstein, G., Reif, J.H., LaBean, T.H.: DNA-templated self-assembly of protein arrays and highly conducive nanowires. Science 301, 1882–1884 (2003)

Operation of a DNA-Based Autocatalytic Network in Serum

Elton Graugnard[1], Amber Cox[1], Jeunghoon Lee[2], Cheryl Jorcyk[3], Bernard Yurke[1,4], and William L. Hughes[1]

[1] Materials Science & Engineering,
Boise State University, Boise, ID 83725 USA
willhughes@boisestate.edu
[2] Chemistry & Biochemistry,
Boise State University, Boise, ID 83725 USA
[3] Biological Sciences,
Boise State University, Boise, ID 83725 USA
[4] Electrical & Computer Engineering,
Boise State University, Boise, ID 83725 USA

Abstract. The potential for inferring the presence of cancer by the detection of miRNA in human blood has motivated research into the design and operation of DNA-based chemical amplifiers that can operate in bodily fluids. As a first step toward this goal, we have tested the operation of a DNA-based autocatalytic network in human serum and mouse serum. With the addition of sodium dodecyl sulfate to prevent degradation by nuclease activity, the network was found to operate successfully with both DNA and RNA catalysts.

1 Introduction

Worldwide, approximately 1.3 million deaths per year are caused by lung cancer [1]. Early detection and diagnosis of cancer can lead to decreased mortality rates, yet current screening methods require significant resources [2]. Recently, microribonucleic acids (miRNAs) have been detected in human blood serum [3]. MicroRNAs are small, single-stranded, non-coding RNAs that are 21-23 nucleotides in length and regulate genes by suppression of messenger RNAs [4,5]. Several miRNAs are amplified in various cancers [6], and expression profiling reveals that miRNA signatures can be used for cancer classification and prognosis.

Current diagnosis technology requires reverse-transcription polymerase chain reaction (RT-PCR) to detect miRNAs in serum [7]. Developments in DNA computing have shown that it is possible to construct metastable DNA-based chemical networks that accept DNA as catalytic inputs and generate output DNA strands whose concentration increases exponentially to produce an easily detectable signal [8]. As miRNAs occur in blood in low abundance, such amplification networks would allow for detection without using PCR. In this study, we report the operation of the DNA-based autocatalytic network reported by

Zhang *et al.* in human blood serum with sodium dodecyl sulfate (SDS). The autocatalytic DNA system accepts the input of either DNA or RNA catalysts and produces output signal strands that generate an easily detectable fluorescence signal.

2 Autocatalytic Network

To assess the feasibility of detecting miRNA in human serum using a DNA-based catalytic network, the entropy-driven autocatalytic system developed by Zhang *et al.* was selected as a test network. Figure 1(a) reproduces the autocatalytic network with the same domain naming convention [8]. In this network, strand **4 2bc** is the autocatalyst that initiates signal strand production by toehold-mediated strand invasion of the substrate complex via the **2b** domain. With an exposed $\bar{\mathbf{3}}$ domain, the fuel strand displaces two autocatalyst strands by two strand invasion processes and forms the waste complex. In each cycle, the amount of autocatalyst strand is doubled, leading to exponential growth of the signal strand. In Fig. 1(b), the released signal strand reacts with a reporter complex to displace a tetrachlorofluorescein (TET) labeled strand. An increase in signal strand concentration is detected by an increase in the TET fluorescence intensity. Zhang *et al.* were able to demonstrate exponential behavior of this autocatalytic network in buffer and in a solution with total mouse liver RNA and rabbit reticulocyte lysate [8], demonstrating successful operation in a complex biological environment.

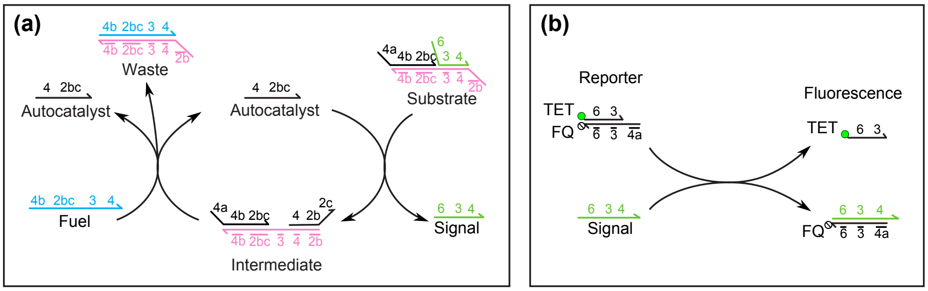


Fig. 1. The DNA-based autocatalytic network reported by Zhang *et al.* [8]. In (a) the autocatalyst initiates the release of the signal strand. The fuel strand displaces both autocatalysts, producing the waste complex. Both released autocatalysts can then initiate new cycles. In (b) the reporter complex consists of a dye-quencher pair in the quenched state. The signal strand from (a) reacts with the reporter complex, displacing the TET dye labeled strand and producing an increase in fluorescence intensity.

3 Network Operation in Serum

To test the autocatalytic system in human serum, the reported DNA sequences of the network were purchased without modification from Integrated DNA Technologies with the same purification processes. Substrate and Reporter complexes were prepared in 1× phosphate buffered saline (PBS) and filtered by polyacrylamide gel electrophoresis to remove excess single-stranded components. Both DNA and RNA versions of the Autocatalyst strand were used to initiate the network.

Whole blood was collected from volunteers and allowed to clot for 30 minutes at room temperature. The clotted solutions were centrifuged at room temperature for 10 minutes leaving the serum as the supernatant, which was extracted to separate vials for storage at -80 °C. In order to ensure successful operation of the autocatalytic network in serum, the use of the ionic detergent sodium dodecyl sulfate (SDS) added to serum was used as a means to suppress nuclease activity without disrupting DNA hybridization. SDS is commonly used to denature proteins, and with 10% SDS, DNA lifetime and hybridization rates in serum are increased [9]. Figure 2 shows the results for operating the autocatalytic network in a solution of 50% human serum, 10% SDS and 0.5×PBS with both DNA, Fig. 2(a), and RNA, Fig. 2(b), catalysts. The data represent the normalized fluorescence intensity of TET dye integrated over one minute intervals with every 75^{th} data point marked with a symbol. For the data shown in Fig. 2(a), the Substrate and Fuel components of the network were present in solution at concentrations of 100 nM, while the Reporter was present at 200 nM. The network was operated with zero added catalyst and with the DNA catalyst added at 10 and 100 nM. The times to half completion for the DNA catalyst were 5.5 min. at 100 nM, 15.6 min. at 10 nM, and 21.7 min. with no catalyst added. For the data shown in Fig. 2(b), the Substrate and Fuel components of the network were present in solution at concentrations of 25 nM, while the Reporter was present at 50 nM. The network was operated with zero added catalyst and with the RNA catalyst added at 2.5 and 25 nM. For the RNA catalyst, the times to half completion were 54.7, 82.1, and 101.3 min. for 25, 2.5, and 0 nM, respectively. The resulting fluorescence versus time data are in qualitative agreement with the results reported previously [8], exhibiting a similar concentration dependence and initial exponential intensity increase. Although the times to half completion for the DNA catalyst are similar to previous results, it should be noted that here the concentrations of all components is 10 times greater. The longer times to half completion for the RNA catalyst are expected for the factor of four reduction in the component concentrations. It should be noted that operation of the autocatalytic network without added catalyst indicates a non-zero leak rate of the system, as observed previously [8]. Methods to reduce this leak rate are currently being studied.

Initial experiments in detecting cancer-related miRNA will be performed in mouse models of lung cancer. To verify that the autocatalytic network can serve as a test system for the detection of miRNA in mouse models, the network was operated in mouse serum. Figure 3 shows the results for autocatalytic network operation in a solution of 50% mouse serum, 10% SDS, and 0.5×PBS with RNA catalyst added at concentrations of 10 and 100 nM, as well as operation with no added catalyst. The measured times to half completion were 13.6, 37.8, and 47.4 min. for 100, 10, and 0 nM, respectively. These half completion times are

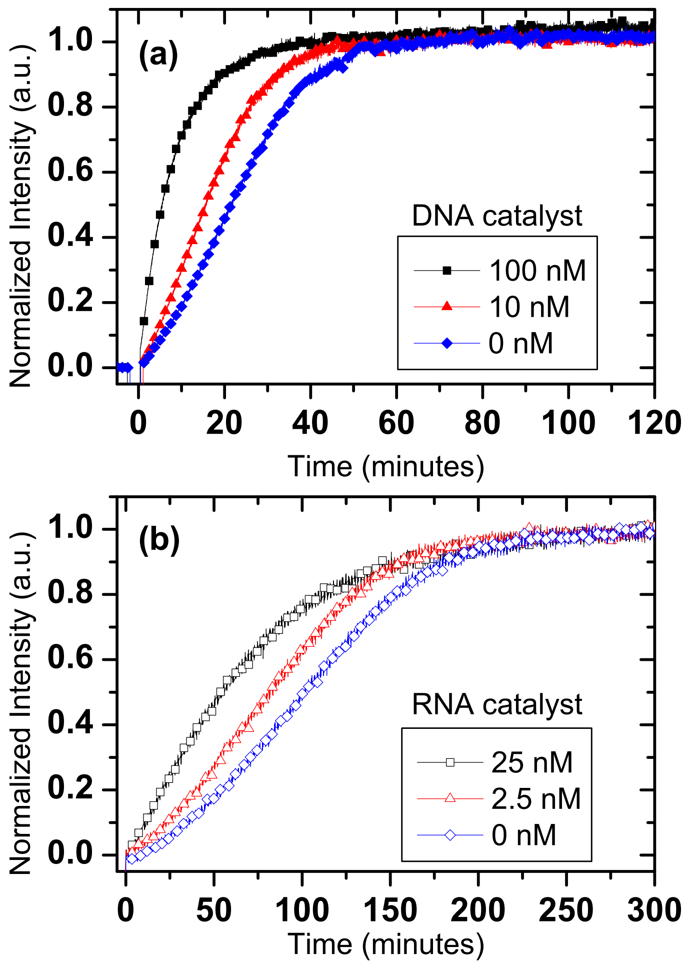


Fig. 2. Autocatalytic network operation in 50% human serum, 10% SDS and 0.5×PBS with (a) DNA and (b) RNA catalysts. Both the DNA and RNA catalysts successfully initiated the autocatalytic network. The observed fluorescence increase and catalyst concentration dependence are in qualitative agreement with the results for network operation in buffer as reported previously [8].

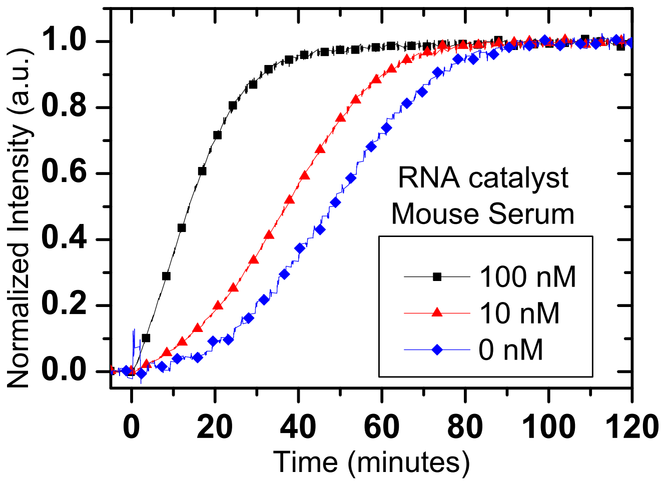


Fig. 3. Autocatalytic network operation in 50% mouse serum, 10% SDS and 0.5×PBS with 100, 10, and 0 nM of RNA catalysts. Network operation in mouse serum was successful with times to half completion comparable to operation in human serum with DNA catalysts.

only slightly longer than those for DNA catalysts in human serum at the same component concentrations, Fig. 2(a), which suggests that test results from mouse models should be readily applicable to human systems.

4 Conclusion

A DNA-based autocatalytic network was successfully operated in 50% human serum, 10% SDS, 0.5×PBS using both DNA and RNA catalysts. Network operation was also confirmed in mouse serum using an RNA catalyst. Operation in serum with 10% SDS was shown to be sufficient to prevent rapid degradation of the network. Times to half completion were similar to those for operation in buffer solution, although the strand concentrations were an order of magnitude higher in the experiments reported here. In all cases, the network exhibited an apparent exponential increase in fluorescence intensity and clear dependence on the catalyst concentration. These results clearly support the feasibility of detecting miRNA in human serum and mouse serum using a DNA-based catalytic network.

Acknowledgments. The authors thank David Y. Zhang, Erik Winfree, Lulu Qian, and William B. Knowlton for valuable assistance with this work and Vivian Lockary, MPH for drawing blood samples. This work was supported by NIH Grant No. P20 RR016454 from the INBRE Program of the National Center for Research Resources, DARPA Contract No. N66001-01-C-80345, and NSF CCF 0855212.

References

1. World Health Organization, Cancer, http://www.who.int/mediacentre/factsheets/fs297/en/print.html
2. American Cancer Society, Cancer Facts and Figures (2009), http://www.cancer.org/Research/CancerFactsFigures/CancerFactsFigures/cancer-facts-figures-2009
3. Chen, X., Ba, Y., Ma, L., Cai, X., Yin, Y., Wang, K., Guo, J., Zhang, Y., Chen, J., Guo, X., Li, Q., Li, X., Wang, W., Zhang, Y., Wang, J., Jiang, X., Xiang, Y., Xu, C., Zheng, P., Zhang, J., Li, R., Zhang, H., Shang, X., Gong, T., Ning, G., Wang, J., Zen, K., Zhang, J., Zhang, C.-Y.: Characterization of microRNAs in serum: a novel class of biomarkers for diagnosis of cancer and other diseases. Cell Res. 18, 997–1006 (2008)
4. He, L., Hannon, G.J.: MicroRNAs: Small RNAs with a big role in gene regulation. Nat. Rev. Genet. 5, 522–531 (2004)
5. Carthew, R.W.: Gene regulation by microRNAs. Curr. Opin. Genet. Dev. 16, 203–208 (2006)
6. Zhang, L., Volinia, S., Bonome, T., Calin, G.A., Greshock, J., Yang, N., Liu, C.-G., Giannakakis, A., Alexiou, P., Hasegawa, K., Johnstone, C.N., Megraw, M.S., Adams, S., Lassus, H., Huang, J., Kaur, S., Liang, S., Sethupathy, P., Leminen, A., Simossis, V.A., Sandaltzopoulos, R., Naomoto, Y., Katsaros, D., Gimotty, P.A., DeMichele, A., Huang, Q., Bützow, R., Rustgi, A.K., Weber, B.L., Birrer, M.J., Hatzigeorgiou, A.G., Croce, C.M., Coukos, G.: Genomic and epigenetic alterations deregulate microRNA expression in human epithelial ovarian cancer. Proc. Natl. Acad. Sci. USA 105, 7004–7009 (2008)
7. Mitchell, P.S., Parkin, R.K., Kroh, E.M., Fritz, B.R., Wyman, S.K., Pogosova-Agadjanyan, E.L., Peterson, A., Noteboom, J., O'Briant, K.C., Allen, A., Lin, D.W., Urban, N., Drescher, C.W., Knudsen, B.S., Stirewalt, D.L., Gentleman, R., Vessella, R.L., Nelson, P.S., Martin, D.B., Tewari, M.: Circulating microRNAs as stable blood-based markers for cancer detection. Proc. Natl. Acad. Sci. USA 105, 10513–10518 (2008)
8. Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318, 1121–1125 (2007)
9. Graugnard, E., Cox, A., Lee, J., Jorcyk, C., Yurke, B., Hughes, W.L.: Kinetics of DNA and RNA Hybridization in Serum and Serum-SDS. IEEE Trans. Nanotechnol. 9, 603–609 (2010)

Triangular Tile Self-assembly Systems

Lila Kari, Shinnosuke Seki, and Zhi Xu

The University of Western Ontario, Department of Computer Science,
London, Ontario, Canada N6A 5B7
{lila,sseki,zhi_xu}@csd.uwo.ca

Abstract. We discuss theoretical aspects of the self-assembly of triangular tiles; in particular, right triangular tiles and equilateral triangular tiles. Contrary to intuition, we show that triangular tile assembly systems and square tile assembly systems are not comparable in general. More precisely, there exists a square tile assembly system S such that no triangular tile assembly system that is a division of S produces the same final supertile. There also exists a deterministic triangular tile assembly system T such that no square tile assembly system produces the same final supertiles while preserving border glues. We discuss the assembly of triangles by triangular tiles and show triangular systems with $\Theta(\log N/\log\log N)$ tiles that can self-assemble into a triangular supertile of size $\Theta(N^2)$. Lastly, we show that triangular tile assembly systems, either right-triangular or equilateral, are Turing universal.

1 Introduction

The basic model of DNA computation by self-assembly has been the one proposed by Adleman [1] and Winfree [9], based on the theory of Wang tiles [7]. In this model, the basic components are *square tiles* with sides painted with "glues", that can stick together to form supertiles if the glues at abutting edges match.

A regular tiling of the plane is a highly symmetric tiling made up of congruent regular polygons. Only three such regular tilings exist: those made up of equilateral triangles, squares, or hexagons. This paper departs from the existing model of self-assembly by investigating, instead of square tiles, the case of *triangular tiles*. We namely discuss the self-assembly by equilateral and right-triangular tile systems.

Our line of investigation follows that started by Winfree [8], who showed how the formation of large structures from certain DNA molecules can simulate Blocked Cellular Automata (BCA), which have the computational power of Turing machines. In 1998, Winfree, Liu, Wenzler, and Seeman [9] designed and experimentally produced two-dimensional DNA crystals by self-assembly. The self-assembly of square tiles was initiated by Adleman [1] who studied the time complexity of a particular case of linear self-assembly. In 2000, Rothemund and Winfree [6] studied the self-assembly of squares at fixed temperature (the threshold that the sum of the strengths of glues of a tile have to surpass, in order for it to "stick" to an existing assembled shape), and showed that in order to deterministically self-assemble an $N \times N$ full square, N^2 different tile types are required at temperature $\tau = 1$ and $O(\log N)$ different tiles suffice at fixed temperature $\tau \geq 2$. In 2001, Adleman, Cheng, Goel, and Huang [2] improved the latter result to $\Theta(\log N / \log \log N)$ different tiles.

In this paper we follow a similar line of inquiry for triangular tiles. In Sect. 2 we introduce the definition of triangular tile assembly systems. In Sect. 3 we compare the square tile assembly systems and triangular tile assembly systems from the point of view of shape complexity and show that the two types of systems are not comparable. In Sect. 4 we discuss the computational power of triangular tile assembly systems and show they are Turing universal. In Sect. 5 we discuss the efficient assembly of triangles by triangular tiles. At the end, we summarize the main results.

2 Definitions

A *triangular tile* is a triangle with each side colored with *glues* from a finite set Σ of glues with associated strengths. In the figures, the strength associated with the glue on a side will be represented by the number of parallel edges along the side. We assume that the shortest side of a triangular tile is of unit length, and assume that a triangular tile cannot be rotated nor flipped over. *Right triangular tiles* are triangular tiles of the shape of right triangles with the right angle pointing to the four possible directions: east, north, west, south as illustrated in Fig. 1(a). More formally, a right triangular tile is a quadruple $(\gamma_1, \gamma_2, \gamma_3, k)$, where $\gamma_i \in \Sigma$ are the glues on the sides of the tile in the counter-clockwise order starting from the longest side, and $k \in \{\mathtt{e}, \mathtt{n}, \mathtt{w}, \mathtt{s}\}$ presents the direction pointed to by the right angle. *Equilateral triangular tiles* are triangular tiles of the shape of equilateral triangles that are either in an upward position or in a downward position as illustrated in Fig. 1(b). More formally, an equilateral triangular tile is a quadruple $(\gamma_1, \gamma_2, \gamma_3, k)$, where $\gamma_i \in \Sigma$ are the glues on the sides of the tile in the counter-clockwise order starting from the horizontal side and $k \in \{\mathtt{u}, \mathtt{d}\}$ presents the upward, respectively downward orientation of the "arrow" represented by the triangle.

A particular glue $\phi \in \Sigma$ denotes the non-interactive glue. The *temperature* $\tau \in \mathcal{R}$ specifies the threshold needed for a tile to stick, as explained below, where $\mathcal{R} = \{0, 1, \ldots\}$ is the set of non-negative real numbers. The *strength function* $g : \Sigma \times \Sigma \to \mathcal{R}$ is defined such that $g(\gamma, \gamma') = g(\gamma', \gamma)$ and $g(\phi, \gamma) = 0$ for all $\gamma, \gamma' \in \Sigma$. In particular, we are interested in the discrete case where τ is an integer, $\Sigma = \Gamma \times \mathcal{N}$ and $g((a, n), (a', n')) = n$ if $a = a', n = n'$ otherwise $g((a, n), (a', n')) = 0$, where Γ is a set of glue labels.

Tiling the plane amounts to mapping tiles onto the lattice of a coordinate system on the plane. The oblique coordinate system $C_{\pi/3}$ whose two axes intersect with the $\pi/3$ angle is the best choice for equilateral triangular tiles. The right triangular tile accords with both rectangular and oblique coordinate systems. The conversion among these coordinate systems can be achieved by affine transformations, which include rotation, scaling, shift, and their compositions.

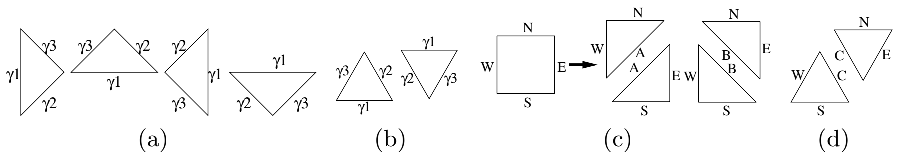


Fig. 1. (a) Four right triangular tiles $(\gamma_1, \gamma_2, \gamma_3, \mathtt{e})$, $(\gamma_1, \gamma_2, \gamma_3, \mathtt{n})$, $(\gamma_1, \gamma_2, \gamma_3, \mathtt{w})$, and $(\gamma_1, \gamma_2, \gamma_3, \mathtt{s})$; (b) Two equilateral triangular tiles $(\gamma_1, \gamma_2, \gamma_3, \mathtt{u})$ and $(\gamma_1, \gamma_2, \gamma_3, \mathtt{d})$; (c) Two ways to divide a square tile into right triangular tiles; (d) Even with the help of affine transformations, squares are divided into two equilateral triangles in single way.

Without going into formal details, and similar to the way these notions were defined for square tiling systems in [9], we can define the notion of *supertile* that can self-assemble, starting from the *seed*, by individual tiles incrementally *sticking* to the current supertile if the sum of the glue strengths at the abutting edges is greater than or equal to the temperature. In latter sections we will study the assembly of *full* triangular supertiles, where "full" means that the pair of common edges of every two adjacent tiles in the supertile has a positive strength.

A *tile assembly system* (TAS) is a tuple $S = (T, s, g, \tau)$, where T is a finite set of tiles of the same kind (either all square, or of the four types of right square triangles in Fig. 1(a), or of the two types of equilateral triangles in Fig. 1(b)), $s \in T$ is a particular supertile called *seed*, g is a strength function, and τ is the temperature. A *final supertile* of a TAS is a supertile st such that there is a supertile sequence $s = st_0, st_1, \ldots, st = st_n$, where st_{i+1} is obtained by sticking one tile to st_i at temperature τ, and no tile can further stick to st. A TAS is *deterministic* if its final supertile is unique regardless of how the self-assembly proceeds starting from the seed.

For an equilateral triangular TAS $S = (T, s, g, \tau)$, we define a corresponding *"flattened"* right triangular TAS $\mathcal{F}(S) = (U, f(s), g, \tau)$, where $U = \{f(t), t \in T\}$, $f(\gamma_1, \gamma_2, \gamma_3, \mathtt{u}) = (\gamma_1, \gamma_2, \gamma_3, \mathtt{n})$, and $f(\gamma_1, \gamma_2, \gamma_3, \mathtt{d}) = (\gamma_1, \gamma_2, \gamma_3, \mathtt{s})$. Informally, a flattened right-triangular system is obtained from an equilateral triangular one by morphing each of the equilateral trianglular tiles into upward pointing, resp. downward pointing right triangular ones.

3 Shape Complexity

We call a shape X*-compatible*, where $X \in$ {square, right triangle, equilateral triangle}, if the region occupied by that shape on the two dimensional plane can be tiled geometrically by X tiles. For example, a triangle is not square-compatible. For a given X TAS, only the assembly of X-compatible shapes is meaningful, and thus, we only consider the assembly of X-compatible shapes in the following discussion.

To compare the final supertiles of two TAS, we not only compare the shape of the final supertiles, but also compare the glues on the border edges, with possible affine transformation on the shape. We call the power of producing certain supertiles the *shape complexity*, and say that TAS of type A have greater or equal power than TAS of type B if every A-compatible final supertile of system assembled by a tile system of type B can be also assembled by some system of type A.

Proposition 1. *Any supertile of triangle-compatible shape can be produced by a non-deterministic triangular TAS of a constant number of tiles or by a deterministic triangular TAS with n tiles, where n is the total number of tiles needed to geometrically assemble it.*

Proposition 1 can be generalized to tiles of other shapes, such as square tiles. In what follows, we only consider deterministic TAS. Since every X-compatible shape can be a supertile produced by a deterministic TAS of type X, where $X \in \{\text{square, right triangle, equilateral triangle}\}$, the shape complexity of tile self-assembly system with different tile shapes is trivially equivalent in regard with compatible shapes. It is sensible to apply certain constraint on the involved systems when comparing shape complexity. In the following we discuss shape complexity under restrictions not only on the shape but also on glues.

A right triangular TAS T is called a *division* of a square TAS S if for any tile s in S, there is a pair of tiles t, t' such that at temperature $\tau \geq 1$ tiles t, t' with $\pi/4$ rotation can produce s; for any tile t in T, there are a tile t' in T and a tile s in S such that at temperature $\tau \geq 1$ tiles t, t' with $\pi/4$ rotation can produce s. By definition, the division of a square TAS may not be unique, and a right triangular TAS can be the division of two different square TAS (see Fig. 1(c)). The number of tiles in the two systems satisfies the inequality $\sqrt{n(S)} \leq n(T) \leq 4n(S)$, where $n(X)$ presents the number of tiles in a TAS X. One may ask the question whether any square TAS can be trivially converted into a triangular TAS by properly dividing each square tile into triangular tiles. The answer is "NO" as shown in the following lemmas.

Lemma 2. *There exists a deterministic square TAS S such that no division of S produces a final supertile of the same shape (with $\pi/4$ rotation).*

Two examples proving this lemma are illustrated in Fig. 2(a), one for $\tau = 2$ and one for $\tau = 3$. In the figure, each tile is numbered in the order of a possible assembly process. For the left-hand-side example system, each of the square tiles s, $1, \ldots, 6$ can be simulated by a pair of right triangular tiles. There are two sticky edges for tile 7, which are on parallel sides of the square tile, each of which is of strength 1. So under $\tau = 2$ the attachment of tile 7 cannot be simulated by successive attachments of two right triangular tiles to assemble the same final supertile. For the right-hand-side example system, by similar reasoning, the attachment of tile 11 cannot be simulated by successive attachments of two right triangular tiles, and thus the assembly stops and fails to grow into a square.

The left-most supertile in Fig. 2(a) has a missing tile in the middle, and we say that it has *"hole"*. More formally, a supertile has no hole if it is full and has no hole geometrically (every region enclosed by tiles is occupied by tiles).

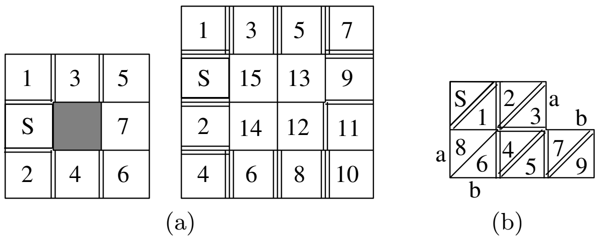


Fig. 2. Examples show that square TAS and triangular TAS are not comparable: (a) two square systems, none of which can be simulated by triangular systems; (b) a triangular system (with $\pi/4$ rotation) that cannot be simulated by any square systems. Each glue, unless mentioned, is unique, and thus, the label is omitted.

Lemma 3. *For any square TAS S at $\tau = 1$, and any square TAS at $\tau = 2$ whose final supertile has no hole, there is a division of S that produces a final supertile of the same shape (with $\pi/4$ rotation).*

By Lemmas 2 and 3, we see that square TAS can be simulated by their division only under certain conditions. Now we discuss the other direction: whether every right triangular TAS can be simulated by a square triangular TAS, assuming that the final supertile is square-compatible.

Since a supertile produced by a TAS may be a building block of a later self-assembly, in this sense, the border glue is as important as the shape of the supertile. In the following lemmas, when comparing two TAS, we ask the final supertile has the same border glues.

Lemma 4. *There exists a deterministic right triangular TAS T such that the final supertile of T is square-compatible but no square TAS produces a final supertile of the same shape (with $\pi/4$ rotation), and that has the same border glues.*

An example of a right triangular TAS such as the one postulated in Lemma 4 is illustrated in Fig. 2(b), where each tile is numbered in the order of a possible assembly process. Note that any square TAS that produces a supertile of the same shape as in Fig. 2(b) must include a square tile with west side glue a, and south glue b. However, if a tile system contains such a tile, its assembly will grow at its north-east corner, and thus, cannot produce a final supertile of the required shape.

Lemma 5. *For any right triangular TAS at $\tau = 1$ whose final supertile is square-compatible, there is a square TAS that produces a final supertile of the same shape (with $\pi/4$ rotation) and keeps the same border glues.*

For every equilateral triangular TAS T, there is a right triangular TAS $\mathcal{F}(T)$ such that the two final supertiles are equivalent up to an affine transformation. We will see that the right triangular TAS in Fig. 7(b) cannot be simulated by an equilateral triangular TAS. So the equilateral triangular TAS are strictly less powerful than right triangular TAS in shape complexity.

The example given in Lemma 4 is a flattened equilateral triangular TAS. In other words, there exists an equilateral triangular TAS which cannot be produced by any square TAS even under affine transformations. By Lemmas 2 and 4, we have the following theorem.

Theorem 6. *The square TAS and the triangular TAS are not comparable in the sense of shape complexity.*

4 Computational Complexity

Several problems on the computational complexity of Wang tile systems were studied by Berger [3] and Robinson [5]. Among them, the *tiling problem* asked whether, given a Wang tile system, one can decide whether or not it can tile the full plane. This was proved undecidable by simulating a Turing machine by a Wang tile system, and then reducing the Halting problem to the Tiling problem. Since the Robinson's Wang tile system which simulates a given TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathtt{B}, F)$ on the input $a_1a_2 \cdots a_n$ can be regarded as a deterministic square TAS S (see Fig. 3), it is undecidable whether a given deterministic square TAS tiles the full plane or not.

Theorem 7. *It is undecidable whether a given deterministic right triangular TAS can tile the full plane.*

The proof is based on the observation that the cooperation between tiles in this Turing machine simulation by S occurs by coordination of either the south edge and the east edge of a square tile, or between the south edge and west edge of a square tile. We can thus construct a triangular division of the square tile system S by dividing every square tile into two right triangular tiles in one of the two ways described in Fig. 1(c) that leaves the cooperating two edges in the same triangular tile. By making the glue of each diagonal edge unique to the square tile that is being divided by it, the resulting right triangular TAS is deterministic, and has at most twice as many tile types as S, that is, $2(n + 2 + 2|\Sigma| + |\delta| + 2|Q||\Sigma|)$. This upper bound can be further improved by $|\delta|$ if we reuse some of the triangular tiles that appear in different square tiles.

This proof technique does not work anymore for equilateral triangle systems, even with the help of affine transformations. This is because when dividing a square into two equilateral triangles as in Fig. 1(d), only one choice exists, e.g., the one that preserves the co-operation between the south and west edges of the original square tile. Nevertheless based on the Robinson's design principle, we can construct a deterministic equilateral triangular TAS which simulates the computation of M on the input $a_1a_2 \cdots a_n$ with at most $2n + 5 + 4|\Sigma| + 2|\delta| + 3|Q||\Sigma|$ (see Fig. 4).

Theorem 8. *For any Turing machine M, there is a deterministic equilateral triangular TAS S at temperature $\tau \geq 2$ such that M does not halt on the blank tape if and only if S tiles the full plane.*

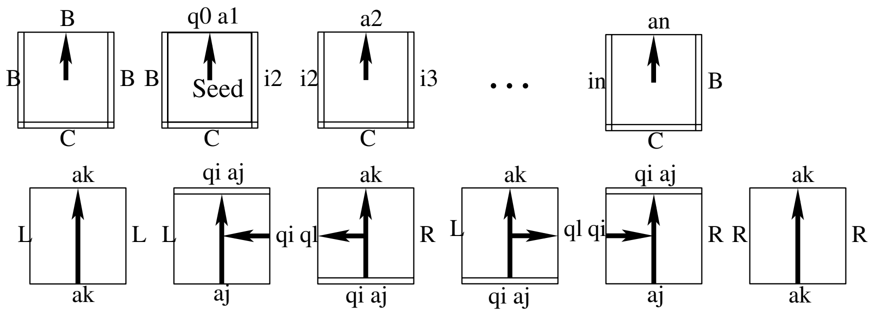


Fig. 3. Tiles of a square TAS S that simulate a given Turing machine with input $\cdots \mathtt{B} q_0 a_1 a_2 \cdots a_n \mathtt{B} \cdots$ at temperature $\tau = 2$. Starting from the seed (second left tile), the first $n+1$ tiles self-assemble the initial configuration. Successive configurations are simulated by addition of rows of tiles, one on top of each other. The i-th row represents the configuration at time i. A transition of the type $\delta(q_i, a_j, \mathtt{L}) = (q_\ell, a_k)$ is simulated by the second and third tiles on the bottom row, while $\delta(q_i, a_j, \mathtt{R}) = (q_\ell, a_k)$ is by the forth and fifth ones. The other two tiles propagates upwards the input letters as well as blank symbols which are not involved in the current transition. The lower half is filled with a filler-tile whose edges are all labeled by $(\mathtt{C}, 2)$ (not shown here).

Proof. The proof idea is a modification of Robinson's construction that crucially uses both the glue-strength and temperature 2 features, as well as the dynamic aspect of self-assembly. Indeed, while the cooperation between the south and west edges of a square tile can be accomplished by a division of a square tile in S into two triangles as in Fig. 1(d) and the previous theorem, the co-operation between the south and east edges cannot. To simulate the behaviour of the square tile system S in this regard, we use two main techniques. The first is to label differently the tiles at the left of the Turing Machine head, versus the ones at the right of it. The second one is to use a different order on which the assembly is build, by using a particular strength 2 glue pattern (rather than within-the-tile edge co-operation) to drive it.

Given a deterministic Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, \mathtt{B}, F)$, we simulate its computation on the input $a_1 a_2 \cdots a_n$ by a deterministic equilateral triangular TAS whose tile set is shown in Fig. 4. Without loss of generality, we can assume that M always moves its head when it transits.

The initial configuration $\cdots \mathtt{B} q_0 a_1 a_2 \cdots a_n \mathtt{B} \cdots$ self-assembles from the seed $(q_0 a_1, B_L, a_2, \mathtt{d})$ with the tiles on the bottom row of Fig. 4 in a straightforward manner as shown in Fig. 5. Each letter is coupled either with the indicator L if the letter is to the left of the head or with R otherwise. Note that the top edges with the TM head or to the left of the head are double-lined, and hence are bound to their matching bottom edges with strength 2. Thus, for instance, the upward alphabet tile with L at its bottom can stick to these top edges without any cooperation so long as their letters match. This is not the case for the edges to the right of the head because their glue strength is 1.

Let us consider the transition $\delta(q_0, a_1) = (q_1, b_1, \mathtt{R})$ first (see Fig. 6(a)). Via the edges with strength 2, the upward alphabet tiles simultaneously stick to the edges

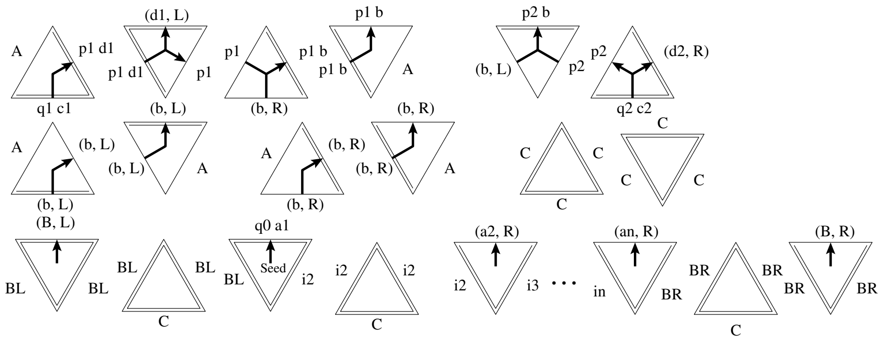


Fig. 4. Tiles of an equilateral triangular TAS which simulates a Turing machine on input $a_1a_2\cdots a_n$. Starting from the seed $(q_0a_1, \mathtt{B}_L, a_2, \mathtt{d})$, the initial configuration $\cdots\mathtt{B}q_0a_1a_2\cdots a_n\mathtt{B}\cdots$ self-assembles using the tiles on the bottom row. Note that tiles to the left of the TM head are marked by L, while the tiles to the right of the TM head are marked by R. This is the case also for the successive configurations. The transition $\delta(q_1, c_1) = (p_1, d_1, \mathtt{R})$ is achieved by the first four tiles on the top row, while $\delta(q_2, c_2) = (p_2, d_2, \mathtt{L})$ is by the other two. The first four tiles in the middle row are used to propagate upwards the letters or the blank symbol which are not involved in the current transition.

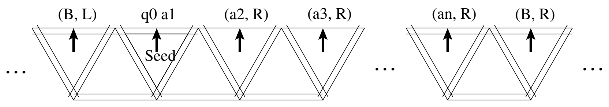


Fig. 5. Simulate the initial configuration.

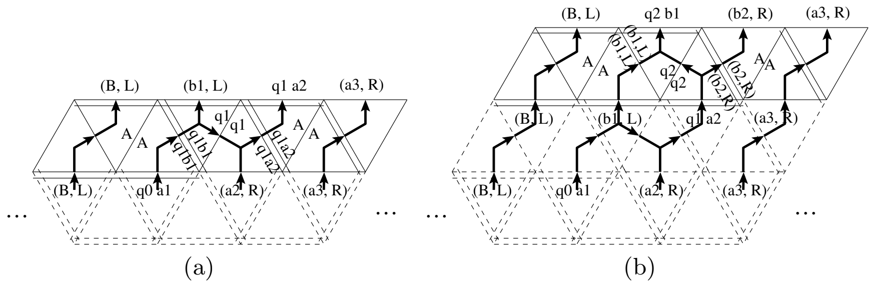


Fig. 6. Simulate the transitions (a) $\delta(q_0, a_1) = (q_1, b_1, \mathtt{R})$ and (b) $\delta(q_1, a_2) = (q_2, b_2, \mathtt{L})$

located to the left of the TM head. The downward alphabet tiles then extend any of these upward alphabet tiles but the one next to the TM head. It is not until an action tile $((q_0a_1, q_1b_1, A, \mathtt{u})$ is stuck to the supertile that the one next to the head is thus extended. The action tile changes the state q_0 and the letter a_1 according to the transition to q_1 and b_1 deterministically, and the downward tile $((b_1, L), q_0a_1, q_1, \mathtt{d})$ branches the letter b_1 coupled with the indicator L up and the state q_1 to the right. Now the merging tile $((a_2, R), q_1a_2, q_1, \mathtt{u})$ can attach by the cooperation of left and bottom edges, and the attachment of its corresponding upward tile immediately follows. The letters to the right of TM head are extended one by one in this manner.

The transition $\delta(q_1, a_2) = (q_2, b_2, \mathtt{L})$ is simulated essentially in the same manner as the previous simulation so that it may suffice to illustrate it as in Fig. 6(b).

This Turing machine simulator consists of at most $2n+5+4|\Sigma|+2|\delta|+3|Q||\Sigma|)$ tiles, where n is the length of the input $a_1 \cdots a_n$. □

Corollary 9. *It is undecidable whether a given deterministic equilateral triangular TAS can tile the full plane.*

5 Self-assembly of Triangles

In this section, we consider deterministic TAS whose final supertile is a upward full triangle with the shortest edge of length N. We call such triangles *N-triangles.*

Proposition 10. *At temperature $\tau = 1$, the minimal number of tile types of a triangular TAS that can assemble an N-triangle is N^2.*

Now we consider the case of temperature $\tau \geq 2$. First we show how to use $2N-1$ triangular tiles to assemble an N-triangle.

Proposition 11. *At temperature $\tau = 2$, there is a triangular TAS of $2N - 1$ tile types that assembles an N-triangle.*

The system is illustrated in Fig. 7(a). Starting from the seed, the bottom line is deterministically assembled. Afterwards, a layer of upward triangular tiles is attached by strength 2 glues on their bottom side, and then a layer of downward triangular tiles is attached by two strength 1 oblique glues. The construction here works for both equilateral triangular tiles and right triangular tiles.

Using a similar technique as that of square tile assembly for $N \times N$ squares [6], the following result follows.

Proposition 12. *There is a right triangular TAS of $O(\log N)$ tile types that assembles an N-triangle.*

The construction of an N-triangle with $2n + 37$ tile types is illustrated in Fig. 7(b), where $n = \lceil \log N \rceil$ and the temperature is $\tau = 2$. Starting from the seed, a supertile with glues that code the integer $(2^n - N + n + 2)/2$ is assembled. Then the tiles simulate the counting up to 2^{n-1} with duplicate copies. For the addition step of the counting, the assembly proceeds from the south-east to the north-west by north and south tiles; for the duplication step, the assembly proceeds from the north-west to the south-east by east and west tiles. This produces the shaded rectangle in Fig. 7(b). This rectangular supertile is then expanded into an N-triangle.

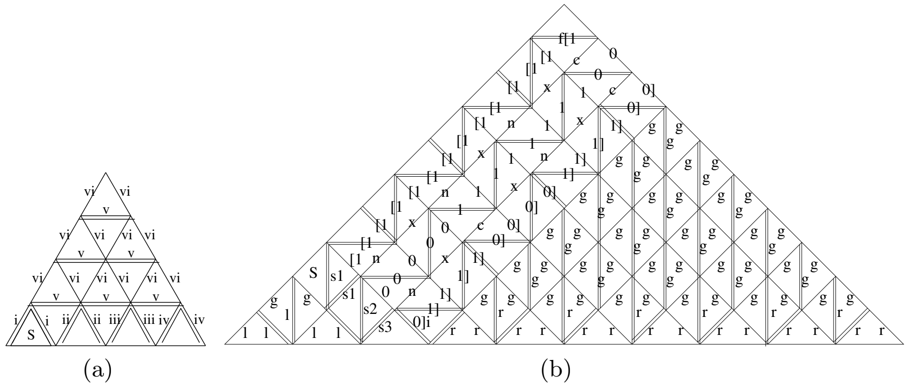


Fig. 7. Triangular TAS that produces a full triangle: (a) An equilateral triangular system of $2N - 1$ tiles for $N = 4$. (b) A right triangular system of $2n + 37$ tiles for $N = 10, n = \lceil \log N \rceil = 4$, where the label on the supertile is omitted for simplicity. Temperature is $\tau = 2$ and S is the seed.

Using the same technique of base conversion as in the square tile assembly [2] the bound on the minimal number of tiles required to assemble an N-triangle can be improved to $O(\log N / \log \log N)$. This is optimal, which is seen by noticing that a square supertile can be assembled by sticking together two right triangular supertiles.

Corollary 13. *There is a right triangular TAS of* $O(\log N / \log \log N)$ *tile types that assembles an* N*-triangle.*

6 Conclusion

The model we used in this paper is at fixed temperature, unit growing (at each step only a single tile sticks to the supertile), and irreversible (supertiles cannot break). There are other possible choices of models. For example, if we allow variable temperature and reversible process as discussed on square tiles [4], then in exactly the same manner to the assembly of squares, one can prove that $O(1)$ tiles are enough to assemble arbitrary large triangle-compatible triangles; in that case the time sequence is of length $O(\log N)$.

Acknowledgements. We thank Dr. David Doty for comments on earlier versions of the paper. This research was supported by The Natural Sciences and Engineering Council of Canada Discovery Grant R2824A01 and Canada Research Chair Award to Lila Kari.

References

1. Adleman, L.: Toward a mathematical theory of self-assembly (1999) (manuscript), https://eprints.kfupm.edu.sa/72519/1/72519.pdf
2. Adleman, L., Cheng, Q., Goel, A., Huang, M.: Running time and program size for self-assembled. In: Proc. 33rd Ann. ACM Symp. Theor. of Comp (STOC 2001), pp. 740–748 (2001)

3. Berger, R.: The undecidability of the domino problem. Mem. Amer. Math. Soc. 66, 1–72 (1966)
4. Kao, M., Schweller, R.: Reducing tile complexity for self-assembly through temperature programming. In: Proc. 7th Ann. ACM-SIAM Symp. Discrete Algorithm, pp. 571–580 (2006)
5. Robinson, R.M.: Undecidability and nonperiodicity for tilings of the plane. Inventiones Math. 12, 177–209 (1971)
6. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares. In: Proc. 32nd Ann. ACM Symp. Theor. of Comp (STOC 2000), pp. 459–468 (2000)
7. Wang, H.: Proving theorems by pattern recognition II. Bell System Technical Journal 40, 1–42 (1961)
8. Winfree, E.: On the computational power of DNA annealing and ligation. In: DNA Based Computers: DIMACS Workshop, pp. 199–221 (1996)
9. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539–544 (1998)

Randomized Self Assembly of Rectangular Nano Structures

Vamsi Kundeti and Sanguthevar Rajasekaran

Department of Computer Science and Engineering
University of Connecticut Storrs, CT 06269, USA
{vamsik,rajasek}@engr.uconn.edu

Abstract. Self assembly systems have numerous critical applications in medicine, circuit design, etc. For example, they can serve as nano drug delivery systems. The problem of assembling squares has been well studied. A lower bound on the tile complexity of any deterministic self assembly system for an $N \times N$ square is $\Omega(\frac{\log(N)}{\log(\log(N))})$ (inferred from the Kolmogrov complexity). Deterministic self assembly systems with an optimal tile complexity have been designed for squares and related shapes in the past. However designing $\Theta(\frac{\log(N)}{\log(\log(N))})$ unique tiles specific to a shape which needs to be self assembled is still an intensive task. Creating a copy of a tile is much simpler than creating a unique tile. With this constraint in mind probabilistic self assembly systems were introduced. These systems have $O(1)$ tile complexity and the concentration of each of the tiles can be varied to produce the desired shape. Becker, et al. [1] introduced a line sampling technique which can self assemble $m \times n$ rectangles, where m is the expected width and n is the expected height of the rectangle. Kao, et al. [2] combined the line sampling technique with binary counters in a novel way to self assemble a supertile which can encode a binary string. This supertile can then be used to produce an $n' \times n'$ square such that $(1-\epsilon)n \leq n' \leq (1+\epsilon)n$ (for some relevant ϵ) with probability $\geq 1-\delta$ for sufficiently large n (i.e., $n \geq f(\epsilon, \delta)$, for some appropriate function f). Doty [3] made the idea of Kao more precise, however the underlying construction is still based on sub-tiles to perform binary counting and division.

In this paper we present randomized algorithms that can self assemble squares, rectangles and rectangles with constant aspect ratio with high probability (i.e. $\Omega(1 - 1/n^{\alpha})$, for any fixed $\alpha > 0$) where n is the dimension of the shape which needs to be self assembled. Our self assembly constructions do not need any *approximation frames* introduced in Kao et al. [2] and hence are much cleaner and has significantly smaller constant in the tile complexity compared to both Kao [2] and Doty [3]. Finally In contrast to the existing randomized self assembly techniques our techniques can also self assemble a much stronger class of rectangles which have a fixed aspect ratio (α/β).

1 Introduction

Self assembly is an autonomous process by which simple nanoscale components assemble into much complex nano structures. Self assembly is ubiquitous in nature in the form of lipid polymers and crystals. Theoretical foundations for understanding how to assemble these nano components into desired shapes is of central importance to nanotechnology. Applications of self assembly span from human health to nanoelectronics. Fundamental limits of optical lithography in circuit fabrication is being addressed by directed self assembly. Gold nanostructures are being used for drug delivery in controlled fashion to treat cancer. It has been proven [4] that DNA crossover molecules can help in building stable structures.

The *tile assembly model* introduced by Winfree [5] has gained importance because of its close proximity to the physical self assembly process. This model is based on the theory of Wang's [6] tiling of planar shapes. Informally this model has unit square tiles with a colored glue attached to each side. Two tiles can stick to each other along an edge only if they have a glue of the same color along the sticking edge. These tiles cannot be rotated during the self assembly process and they can only be translated. Given a target shape the goal is to design these tiles with colored glues so that they can be assembled into the target shape uniquely. The efficiency of the tileset is determined by the number of unique tiles used in the tileset. Initial work in this area focussed on the design of tilesets which can assemble the target shape exactly [7] [8] [9] [10]. Efficient tilesets were designed for self assembling squares by Rothemund, et al. [7] and Adleman, et al. [8]. Also a lower bound of $\Omega(\frac{\log(n)}{\log(\log(n))})$ on the number of unique tiles to self assemble a square has been established by Rothemund, et al. [7]. This lower bound is dictated by the Kolmogrov complexity. In all of these tilesets the information about the dimension is encoded in the tiles. As the dimension of the target shape increases the number of unique tiles also increases. The design of a large number of unique tiles is an intensive task. However creating a copy of an existing tile is a straightforward process in the laboratory. This observation has led to a new paradigm of probabilistic self assembly systems. In probabilistic self assembly systems the information about the dimension is encoded in the concentration of the tiles. Thus the number of unique tiles in these systems is reduced considerably. However the price we pay in the probabilistic self assembly systems is that the final structure may not have the exact dimensions we are targeting. The difference between the target dimension and the assembled dimension is bounded by a certain probability which is of our interest.

The idea of a probabilistic self assembly was introduced by Becker, et al. [1]. They showed how to self assemble squares and rectangles which are expected to have dimensions close to the target dimensions. The techniques introduced by Becker, et al. suffered from high variance on the dimensions achieved. This shortcoming was addressed by Kao, et al. [2].

Later Doty [3] independently considered the problem of making the idea of Kao [2] more precise. Doty [3] showed that the line sampling estimate of n (the dimension of the square) can be made precise by estimating smaller parts of the binary representation of n. However his construction still needs to encode a binary number on the tiles to self assemble a $n \times n$ square. The constructions of both Kao [2] and Doty [3] are not simple and involve sub-tilesets to perform binary addition and division – thus have large hidden constants. In contrast one of the major results of our paper is to show how to construct simple tilesets without the need to have sub-tilesets to perform binary addition and division. Our algorithms can self assemble squares and rectangles with dimensions close to the target with high probability. Also, all the existing randomized techniques only address self assembly of squares. In this paper we also give constructions that can self assemble rectangles with constant aspect ratio (α/β) and squares are a special case of these rectangles. These rectangles have dimensions close to target dimensions with high probability. Our rectangle constructions are applicable in the deterministic context as well.

In this paper we use n to denote the target dimension that needs to be assembled and n' to denote the dimension of the final assembled structure.

2 Basics of the Tile Assembly Model

The tile model $t = (N(t), S(t), W(t), E(t)) \in \Sigma^4$ used in the assembly is a square shaped Wang tile with glues/symbols (from alphabet Σ) attached to each of the four sides. $N(t), S(t), W(t), E(t)$ denote symbols attached to the north, south, west and east directions, respectively. Note that these tiles cannot be rotated and hence $(\sigma_n, \sigma_s, \sigma_w, \sigma_e) \neq (\sigma_e, \sigma_w, \sigma_n, \sigma_s)$. A *self assembly system* is characterized by a five tuple $\langle T, s, \tau, G, P\rangle$. Symbol T denotes the *tileset* $T \subset \Sigma^4$ and is a collection of unique tiles used in the assembly process. The tile s is called the *seed tile* which is fixed at a particular location to initiate the assembly process. The symbol τ is a positive integer that indicates the *temperature* of the model. The function $G : \Sigma \times \Sigma \rightarrow \{0, 1, \dots \tau\}$ is called the *bond strength* function. For any two symbols $x, y \in \Sigma$ the function $G(x, y)$ is defined as follows. The symbol ϵ denotes a empty symbol.

$$G(x,y) = \begin{cases} 0 & \text{If } (x \neq y) \vee (x = \epsilon \vee y = \epsilon) \\ k \in \{1, 2, \dots \tau\} & \text{If } x = y \neq \epsilon \end{cases}$$

Informally the function G associates a glue strength with each symbol in Σ. And a *bond* can only be formed between the glues which are of the same type/symbol. The function P is the probability distribution associated with a subset of tiles in T. Since we study the self assembly of only planar shapes we consider only self assembly models which have a temperature $\tau = 2$.

Theoretically we can think of the self assembly as a process that occurs on an infinite 2-dimensional integer grid. We define a null tile ϕ as follows $\phi := (\epsilon, \epsilon, \epsilon, \epsilon)$. The state of the self assembly process at any stage is represented by a mapping $C' : \mathbb{Z} \times \mathbb{Z} \rightarrow \{T \cup \phi\}$. If $C'(x, y) = \phi$ it means that there is no tile placed at position (x, y) on the integer grid. We define a set $C := \{(x, y)|C'(x, y) \neq \phi\}$ as the *configuration* of the self assembly system. It is assumed that the seed tile s is placed initially at position $(0, 0)$ and null tile ϕ is placed on all the grid points except $(0, 0)$. Thus initial configuration of the self assembly system is $C = \{(0, 0)\}$ and $C'(0, 0) = s$. Given a configuration C we say that a tile $t \in T$

is *attachable* to C at (x, y) only if both the following constraints (**C1** and **C2**) hold.

Let $g_1 = G(N(t), S(C'(x, y + 1))) + G(S(t), N(C'(x, y - 1)))$
Let $g_2 = G(W(t), E(x - 1, y)) + G(E(t), W(x + 1, y))$

$C'(x, y) = \phi$ (**C1**: position (x, y) is empty)
$g_1 + g_2 \geq \tau$ (**C2**: neighbour tiles should provide enough strength)

Informally both these constraints mean that a tile $t \in T$ can be assembled to a configuration C at position (x, y) only if that position is empty and it could receive a *glue strength* of at least τ from the neighbouring tiles at position (x, y). If a tile $t \in T$ is *attachable* to C at (x, y), then the configuration C moves to a new configuration C_1 such that $C_1 = C \cup \{(x, y)\}$ and $C'(x, y) = t$. In the self assembly literature a configuration C is also referred to as a *supertile* C. We can describe the self assembly process with a directed acyclic graph $D_g = (V, E)$, where V is set of all supertiles and E is set of directed edges. There is a directed edge (C_1, C_2) between two supertiles only if $\exists t \in T$ which is *attachable* to supertile C_1 to get the supertile C_2. The *in-degree* of the initial supertile C (i.e., the seed tile s at position $(0, 0)$) is zero. The self assembly process continues to move from one supertile to the other with the help of some *attachable* tile $t \in T$. The assembly process stops at a supertile C_t if $\nexists t \in T$ which is *attachable* to C_t. Such a supertile is called a *terminal* supertile. A tile system $\Gamma = \langle T, s, \tau, G, P \rangle$ is said to produce a shape Υ *uniquely* if all the *terminal* supertiles have exactly the same shape (including dimensions) as Υ. Deterministic assembly systems focus on the construction of tilesets which can *uniquely* assemble a give shape with minimum *tile complexity.*

2.1 Probabilistic Self Assembly

Probabilistic self assembly $\langle T, s, \tau, G, P \rangle$, in contrast to deterministic assembly, can produce terminal supertiles which correspond to multiple shapes. However one of these shapes (close to the target shape) is produced with high probability. Note that the probability distribution function P for deterministic self assembly is uniform $P[t] = 1/|T|, \forall t \in T$. However in probabilistic self assembly not all the tiles will have the same probability. The probability of a tile in practice can be changed by changing the concentration of the tiles. The information about the dimension of the shape to be assembled is encoded in the concentration of some of the tiles in T.

2.2 Our Results

We summarize our results as follows. First we introduce a new sampling technique based on the sum of random variables which follow geometric distribution. We then use this to show how to construct a supertile which can encode a binary number n' on the tiles such that $|n' - n| < \epsilon n$ with high probability

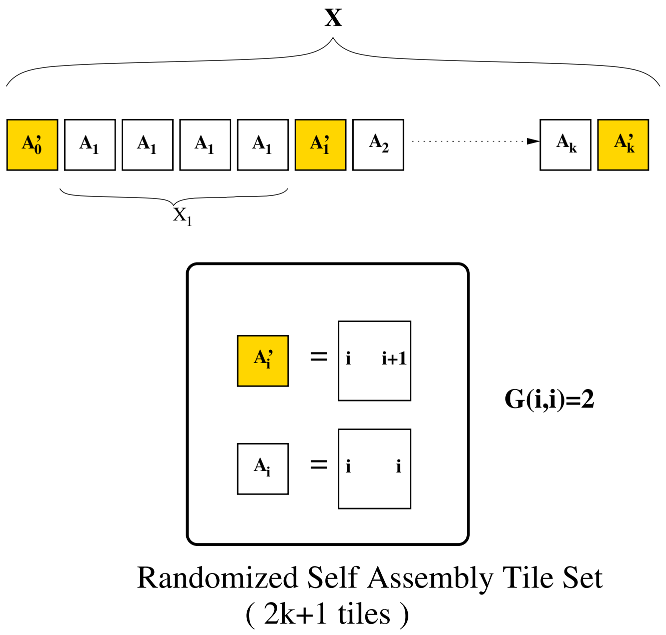


Fig. 1. Randomized self assembly multiple geometric distributions

(i.e., $\Omega(1 - 1/n^\alpha)$, for any fixed $\alpha > 0$). Next we introduce a new self assembly sampling technique called the *staircase sampling*. We then show how this sampling scheme can be used to self assemble squares and rectangles with constant aspect ratio such that $|n' - n| < \epsilon n$ with high probability.

3 Randomized Self Assembly with a Sum of Geometric Distributions

The idea of assigning probabilities to various tile types was considered by Becker, et al. [1]. They introduced a probabilistic tile system consisting of two tiles $A = (\epsilon, \epsilon, \sigma, \sigma)$, $A' = (\epsilon, \epsilon, \sigma, \epsilon)$ and a seed tile $s = (\epsilon, \epsilon, \epsilon, \sigma)$. The glue strength function and the probability distribution of the tiles are defined as follows. $G(\sigma, \sigma) = 2, P[A'] = p$ and $P[A] = 1 - p$. The seed tile s is placed at $(0, 0)$. The self assembly grows a line from left to right. The tile A helps to increase the length of the line since $G(W(A)) = G(E(A)) = 2$. However tile A' terminates the assembly process since $G(E(A')) = 0$. We can associate a random variable L corresponding to the length of the assembled line. It is clear that $E[L] = 1/p$ since L follows a geometric distribution. So if we choose the concentration of tile A' such that $P[A'] = p = 1/n$, then the expected length of the self assembled line is n. By adding a constant number of tiles to this tileset it is also possible to self assemble an $n \times n$ square with expected dimensions. However this line sampling technique suffers from high variances on the dimensions.

Kao, et al. [2] have focused on reducing the variance by introducing an extra tile $A_1 = (\epsilon, \epsilon, \sigma, \sigma)$ into the previous tile set. Note that the tile A_1 has exactly the same definition as A, however it has a different color (say red). Kao, et al. associated a random variable R that corresponds to the number of red tiles (i.e., of type A_1) in a self assembled line of length L. It is easy to see that the random variable R follows a binomial distribution. Since R follows a binomial distribution, Kao, et al. bounded the quantity L/R by applying Chernoff bounds. However Kao, et al.'s tileset needs tiles for computing the values of L, R and L/R. This means that we need sub-tilesets to perform binary addition and division. One of the contributions of our paper is to show that the same high probability bounds can be derived in a much simpler manner by considering a random variable that is expressed as the sum of multiple geometrically distributed random variables. We now give more details of our constructions. Theorem 1 is already known and gives the Chernoff bounds for the sum of random variables that follow a geometric distribution.

Theorem 1. *If $X = \sum_{i=1}^{k} X_i$ is the sum of k independent geometrically distributed variables then for any $0 < \epsilon \leq 1$, $P[X \geq (1+\epsilon)\mu] \leq e^{-\mu\epsilon^2/3}$ and $P[X \leq (1-\epsilon)\mu] \leq e^{-\mu\epsilon^2/2}$ where $\mu = E[X]$.*

Our randomized self assembly technique uses multiple geometric distributions. Consider the line sampling technique introduced earlier which has only two unique tiles A and A'. The sampling line continues to grow with tile A until terminated by tile A'. We now introduce tiles $A_1, A_2 \ldots A_k$ and $A'_0, A'_1, A'_2 \ldots A'_k$. The growth of tile A_i is terminated by tile $A'_{i+1}, 1 \leq i \leq k$. The tile A'_0 denotes the seed tile. In contrast to line sampling we now have k independent self assembling stages terminated by tile A'_{k+1}. Figure 1 further clarifies our idea. We now associate a random variable X corresponding to the length of the terminal self assembled line. We also associate the random variables $X_i, 1 \leq i \leq k$ corresponding to the length of the self assembly between tiles A'_{i-1} and A'_i. Clearly $X = \sum_{i=1}^{k} X_i + (k+1)$ and each of the random variables X_i follows a geometric distribution with a mean of $1/p$ where p is the probability of success. We now prove Theorem 2 based on our randomized self assembly technique. Later we will show how to add a constant number of tiles to this sampling technique to actually encode the binary value on to the tiles. Note that n' and n are used to denote the length of the self assembled line with sampling and the length of the target/required line, respectively.

Theorem 2. *For any given $\epsilon \in (0,1)$ and a positive integer n, the multi geometric sampling technique can self assemble a line of length n' such that $|n' - n| < \epsilon n$ with a probability $\Omega(1 - 1/n^{\alpha})$ for any $\alpha \leq \frac{n\epsilon^2 \log(e/2)}{3log(n)}$.*

Proof. The length of the terminal self assembled line is n' and the target length is n. We have associated a random variable X corresponding to n'. Since $X = \sum_{i=1}^{k} X_i + (k+1)$ and each X_i follows a geometric distribution we can apply Chernoff bounds in Theorem 1 as follows.

$\mu = E[X] = \sum_{i=1}^{k} E[X_i] + (k+1) = k/p + (k+1)$

Let S_1 be the event that $X \geq (1+\epsilon)\mu$;

Let S_2 be the event that $X \leq (1-\epsilon)\mu$;

$\rightarrow \quad P[S_1] \leq e^{\frac{-\mu\epsilon^2}{3}}$ (By Chernoff bounds in Theorem 1)

$\rightarrow \quad P[S_2] \leq e^{\frac{-\mu\epsilon^2}{2}}$ (By Chernoff bounds in Theorem 1)

$\rightarrow \quad P[S_1 \cup S_2] \leq P[S_1] + P[S_2]$ (Union rule)

$P[S_1] + P[S_2] \leq 2e^{\frac{-\mu\epsilon^2}{3}}$

$\rightarrow \quad P[\overline{S_1} \cap \overline{S_2}] \geq 1 - \frac{2}{e^{(\mu\epsilon^2)/3}}$

Now select $p = \frac{k}{n-(k+1)} \rightarrow \mu = n$

$\rightarrow P[\overline{S_1} \cap \overline{S_2}] \geq 1 - 2/e^{(n\epsilon^2)/3} > 1 - 1/n^{\alpha}$ $(\alpha \leq \frac{n\epsilon^2 log(e/2)}{3log(n)})$

Note $\overline{S_1} \cap \overline{S_2} = \{X : |X - \mu| < \epsilon\mu\}$

$\rightarrow \quad P[|n' - n| < \epsilon n] = \Omega(1 - 1/n^{\alpha}).$ □

3.1 Supertile to Encode a Binary String

We now show how our random sampling technique can be used to self assemble a supertile which can encode binary string of log(n) bits that has a value of n'. Building a supertile that can encode a given binary string is fundamental to the construction of several shapes. As shown by Rothemund, et al. [7], given a supertile encoded with a specific binary string, we can build an $n \times n$ square by adding only a constant number of tiles to the tileset. Deterministically optimal tilesets of size $\Theta(\frac{\log(n)}{\log(\log(n))})$ exist for building $n \times n$ squares (see [8]). In [2] Kao, et al. gave a tile assembly that can encode any binary value n' on the tiles. In Kao, et al.'s construction the length of the sampling line does not have a direct correspondence to n' and requires a sub-tile assembly which performs division. In contrast, our multi-geometric sampling technique can achieve equivalent (proved in Theorem 2) stochastic properties for encoding n' in binary with a much simpler supertile construction. Figure 2 gives the overview of how this encoding is done with just a constant (7) number of tiles. The basic idea behind this binary encoding is to integrate a binary counter to the self assembling sampling line by modifying the binary counter tileset presented in [7]. Figure 2 shows the self assembly of a supertile with two geometrically distributed variables when the random variable $X = n'$ takes a value of 13. In this example, $X_1 = 6$ and $X_2 = 4$. The green colored tile in Figure 2 represents the seed tile. The tiles on the left part of the tileset vary with the number of variables used in the sampling process. If we use k geometrically distributed variables, then this would have $2k + 1$ (in this example $k = 2$) unique tiles. The right part of the tileset is independent of the number of variables and will always have 7 tiles in it. The binary value of n' is encoded on the tiles along the right corner of the supertile from top to bottom. We now summarize the discussion in this section with Theorem 3.

Theorem 3. *For any given $\epsilon \in (0,1)$ and a positive integer n there exists a tile system with constant tiles to encode a binary string of value n' such that $|n' - n| < \epsilon n$ with a probability of $\Omega(1 - 1/n^{\alpha})$ for any $\alpha \leq \frac{n\epsilon^2 \log(1/2)}{3log(n)}$.*

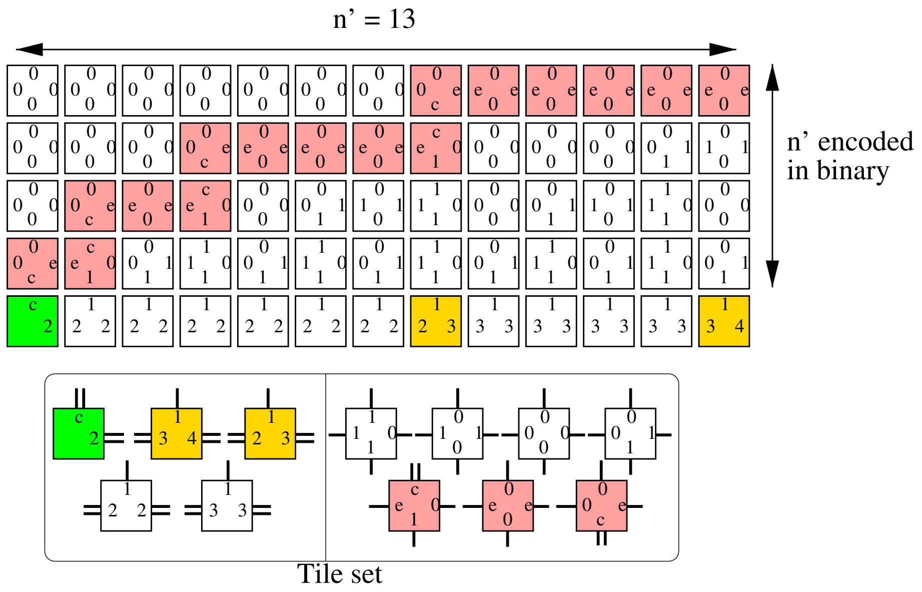


Fig. 2. Self assembly of the supertile encoding a binary string on the tiles. The strength function G of a symbol is illustrated by the bars attached to the tiles. For example strength of the symbol c, $G(c, c)$ is indicated by two bars.

4 New Idea of Staircase Sampling

We now introduce our idea of *staircase sampling*. This technique helps in the randomized self assembly of rectangular shapes with a constant aspect ratio (α/β). The idea of randomized self assembly of rectangles was introduced by Becker, et al. [1]. Their idea generalizes the line sampling described in Section 3 for two dimensions (north, east). They associate two random variables M and N to these sampling lines to self assemble a $m \times n$ rectangle where m and n are the expected height and width, respectively. Since these are only expected dimensions their technique suffers from high variances on these dimensions. Also their technique will not work in self assembling much stronger classes of shapes such as a class of rectangles with constant aspect ratio. Note that the random variables M and N have a geometric distribution. If p_n and p_e are the probabilities of success along north and east, then the probability of producing a $m_1 \times n_1$ rectangle by the self assembly is $P[M = m_1, N = n_1] = (p_n)^{m_1}(p_e)^{n_1}$. This will remain the same even if the aspect ratio is a constant (i.e. $m/n = \alpha/\beta$), since both the random variables are assumed to be independent in their technique.

Let $R(\alpha, \beta) := \{r(x, y) : x/y = \alpha/\beta\}$ denote a class of rectangles with a constant aspect ratio. Given the dimensions of any rectangle $r(n\alpha, n\beta) \in R(\alpha, \beta)$ our aim in this section is to show that we can self assemble $r(n'\alpha, n'\beta) \in R(\alpha, \beta)$ such that n' and n are very close with high probability. The key idea behind our sampling technique is to sample along the staircase in contrast to the existing

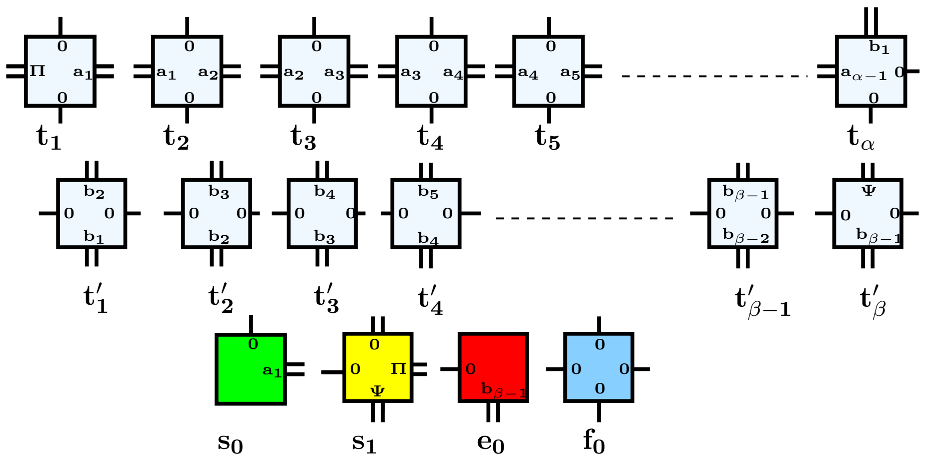


Fig. 3. General tileset for staircase sampling to self assemble a rectangle of aspect ratio α/β

techniques that sample along a straight line. We now show how to construct the tileset for staircase sampling.

4.1 Discussion on the Tileset for Staircase Sampling

We introduce $\alpha + \beta - 1$ unique tiles corresponding to the aspect ratio α/β of the rectangle we would wish to self assemble. Please see Figure 3 to follow this discussion. All the symbols in Figure 3 $\Pi, \Psi, a_1, a_2 \ldots a_{\alpha-1}$ and $b_1, b_2 \ldots b_{\beta-1}$ have a glue strength of 2. The tiles which do not carry any symbol along sides are assumed to have a dummy symbol ω which has a glue strength of 1 (i.e. $G(\omega, \omega) = 2$). With this definition of glue strength function G, the tiles $t_1, t_2 \ldots t_\alpha$ can deterministically self assemble into a horizontal line of length α ending with tile t_α. Notice that tile t_α has a symbol b_1 on the top. Since the symbol b_1 has a glue strength of 2 tiles $t'_1, t'_2 \ldots t'_{\beta-2}$ deterministically self assemble into a vertical line of height $\beta-2$. Till this point the self assembly process is deterministic. However after the tile $t'_{\beta-2}$ gets attached into the vertical line, we now have two tiles ($t'_{\beta-1}$ and e_0)which can get attached on top of tile $t'_{\beta-2}$. In the first case if the tile $t'_{\beta-1}$ gets attached on top of tile $t'_{\beta-2}$ then the self assembly can progress further. This is because the tile $t'_{\beta-1}$ has a symbol Ψ on top which has a bond energy of 2. However in the second case if the tile e_0 gets attached on top of tile $t'_{\beta-2}$ the self assembly process will not grow further vertically since the tile e_0 has no symbol on the top. The filler tile f_0 gets assembled into places which can provide a glue strength of 2 and completes the rectangle.

To further illustrate the idea consider the tileset shown in Figure 4 to self assemble a rectangle with aspect ratio 4/3 ($\alpha = 4, \beta = 3$). Figure 5 shows the

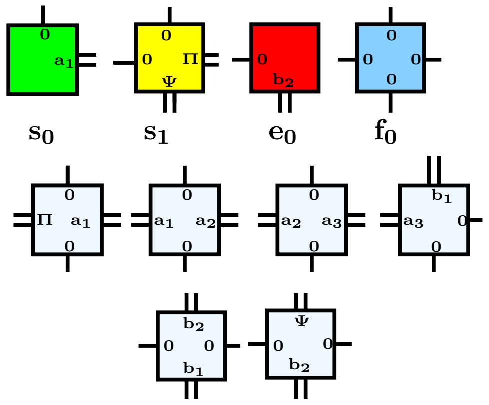


Fig. 4. Tile set for self assembling a rectangle with aspect ratio 4/3

actual assembly process. Notice that the tiles t'_2 and e_0 play a key role in either increasing the width (and height correspondingly) or stopping further growth in the dimension of the rectangle. In general we attach a probability of *success* (p) and *failure* $(1-p)$ to the tiles $t'_{\beta-1}$ and e_0, respectively. We state the following Theorem 4 relevant to the discussion in this section.

Theorem 4. *Given any three integers $\alpha, \beta, n > 1$ the staircase sampling technique can self assemble a rectangle with an expected width and height of $n\alpha$ and $n\beta$ respectively.*

4.2 Generalization of Staircase Sampling with Multiple Variables

We now apply the multi geometric distribution technique introduced in Section 3 to staircase sampling to derive high probability bounds on the dimensions of the terminal self assembled rectangle. We generalize the tileset for staircase sampling in Figure 3 by introducing an extra subscript for all the symbols and tiles. If we choose to have k geometric distributions, then for $1 \le i \le k$ we would have symbols Φ_i, Ψ_i,$a_{i1}, a_{i2} \dots a_{i\alpha}$, $b_{i1}, b_{i2} \dots b_{i\beta-1}$ on the tiles $t_{i1}, t_{i2} \dots t_{i\alpha}$, $t'_{i1}, t'_{i2} \dots t'_{i\beta-1}$ similar to the construction in Figure 3. Also notice that previously we used the tile e_0 (in Figure 3) to stop growth in the dimensions of the rectangle. However in this case we need to introduce k such tiles $(e_0, e_1 \dots e_{k-1})$ with an objective of moving from distribution i to distribution $i+1$. This is similar to the role played by the tiles A'_i in Section 3. To accomplish this task we need to modify tiles $e_{j-1}, 1 \le j \le k-1$ to carry a symbol of Π_j along their right side. Finally we introduce random variables $X_i, 1 \le i \le k$ to indicate the number of tiles of type $t'_{i\beta-1}$ during the i^{th} stage of the multi geometric sampling. Let $X := \sum_{i=1}^{k} X_i$. Clearly if $X = t$ then the final self assembled rectangle will have dimensions $t\alpha \times t\beta$. With this we state the following Theorem 5.

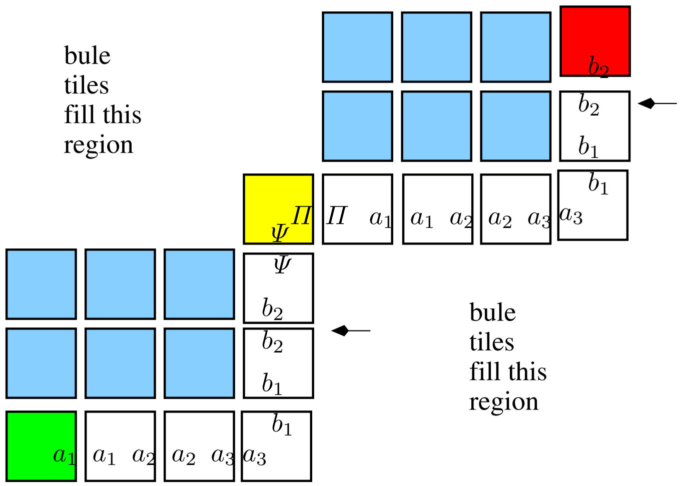


Fig. 5. Self assembly of the rectangle with the tileset in Figure 4

Theorem 5. *For any given $\epsilon \in (0,1)$, and integers $n, \alpha, \beta > 1$, the multi geometric sampling combined with stair case sampling can self assemble a $n'\alpha \times n'\beta$ rectangle such that $|n' - n| < \epsilon n$ with a probability $\Omega(1 - 1/n^q)$ for any $q \leq \frac{n\epsilon^2 \log(e/2)}{3\log(n)}$* . □

A direct corollary when $\alpha = \beta$ self assembles a square of dimension n' such that $|n' - n| < \epsilon n$ with high probability. This is equivalent to what Kao, et al. derived in [2] without the need of any supertile that encodes a binary string. Thus we can conclude that combining multiple geometric variables with stair case sampling lets us derive high probability bounds for self assembling rectangles with constant aspect ratio.

5 Conclusion

In this paper we have presented new constructions for probabilistic self assembly and derived high probability bounds on the dimensions of the terminal supertile. We conclude that probabilistic self assembly with a sum of random variables following geometric distribution is helpful in simpler and cleaner self assembly constructions in contrast to the existing techniques presented in Kao et al. [2]. We also have introduced a new idea of *staircase sampling* which is useful in assembling rectangles with constant aspect ratio. These shapes cannot be assembled with the existing probabilistic self assembly techniques.

6 Future Work

Probabilistic self assembly is an exciting area of further research, especially because of the fact that we need only $\Theta(1)$ tiles to self assemble shapes which are close to the target shapes. One future research will include the investigation of how the current techniques to self assemble squares and rectangles can be combined to self assemble Manhattan shapes within the $\Theta(1)$ tile complexity. Another direction of research would be to explore probabilistic constructions of non-rectangular regular polygons.

Acknowledgements. This work has been supported in part by the following grants: NSF 0829916 and NIH 1R01GM079689-01A1.

References

1. Becker, F., Rapaport, I., Rémila, É.: Self-assemblying classes of shapes with a minimum number of tiles, and in optimal time. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 45–56. Springer, Heidelberg (2006)
2. Kao, M.-Y., Schweller, R.T.: Randomized self-assembly for approximate shapes. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 370–384. Springer, Heidelberg (2008)
3. Doty, D.: Randomized self-assembly for exact shapes. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 85–94. IEEE, Los Alamitos (2009)
4. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of dna sierpinski triangles. PLoS Biology 2(12) (2004)
5. Winfree, E.: Algorithmic self-assembly of dna. dissertation (ph.d.), california institute of technology (1998), `http://resolver.caltech.edu/CaltechETD:etd-05192003-110022`
6. Wang, H.: An unsolvable problem on dominoes. Technical Report BL-30 (1962)
7. Rothemund, P.W.K., Winfree, E.: Program-size complexity of self-assembled squares. In: ACM Symposium on Theory of Computation (STOC), pp. 459–468 (2000)
8. Adleman, L., Cheng, Q., Goel, A., Huang, M.: Running time and program size for self-assembled squares. In: Annual ACM Symposium on Theory of Computing, pp. 740–748 (2001)
9. Aggarwal, G., Cheng, Q.I., Goldwasser, M.H., Kao, M., De Espanes, P.M., Schweller, R.T.: Complexities for generalized models of self-assembly. SIAM Journal on Computing 34(6), 1493–1515 (2005)
10. Kao, M., Schweller, R.: Reducing tile complexity for self-assembly through temperature programming. In: Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 571–580 (2006)

Design of a Functional Nanomaterial with Recognition Ability for Constructing Light-Driven Nanodevices

Xingguo Liang[1,*], Toshio Mochizuki[1], Taiga Fujii[1], Hiromu Kashida[1], and Hiroyuki Asanuma[1,2,*]

[1] Department of Molecular Design and Engineering, Graduate School of Engineering, Nagoya University, Chikusa, Nagoya 464-8603, Japan
[2] Core Research for Evolution Science and Technology (CREST), Japan Science and Technology Agency (JST), Kawaguchi, Saitama 332-0012, Japan
{liang,asanuma}@mol.nagoya-u.ac.jp

Abstract. An artificial macromolecule (foldamer) was designed as a novel nanomaterial with the backbone of phosphodiester and the side chain of functional molecules and nucleobases. The functional molecules tethered on D-threoninol and the nucleosides on D-ribose can be lined up with any sequence and ratio by using standard phosphoramidite chemistry. The nucleobases that form Watson-Crick base pairs provide the sequence recognition which is required for constructing complicate nanostructures. The multiple functional molecules give applicable and advanced functions such as photoresponsiveness when azobenzenes were used. Unexpectedly, a stable double helix was formed even in the case that the ratio of azobenzene molecules and base pairs was as high as 2:1. More interestingly, this artificial duplex showed high sequence specificity: the stability decreased greatly when a mismatched base pair was present. Furthermore, the formation and dissociation of the constructed artificial duplex were reversibly and completely modulated with light irradiation. By using this new nanomaterial, a variety of functional nanostructures and nanodevices are promising to be designed.

Keywords: nanodevice, nucleobase, photoregulation, hybridization.

1 Introduction

In recent years, numerous defined nanoconstructs and nanodevices have been constructed by self-assembly using nucleic acids, protein, or artificial macromolecules including foldamers [1-5]. The constructed nanostructures have great potential to be used in a large variety of nanotechnology fields. However, there are few designs to give applicable functions because the structure is hardly regulated once it is formed. Another reason is that functional molecules cannot be easily introduced to the desired position of nanodevices as will. On the other hand, due to the development of organic chemistry, numerous excellent molecular-level structures have been constructed that

[*] Corresponding authors.

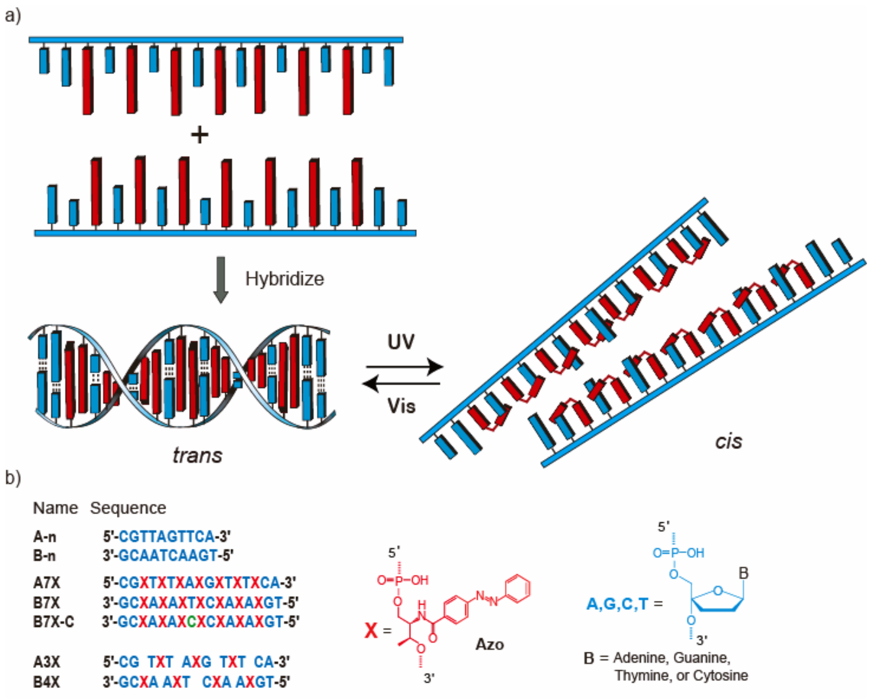
can be switched, rotated, and directionally driven in response to stimuli [6]. However, these functions are also difficult to be designed to work for us because the nanodevices are individually designed and lack the diversity required for specific molecular recognition. In addition, most of these nanomachines are operated in organic solvents but not in water for biological applications [6]. A combination of high functionality and delicate structure is highly required for future applications such as building intelligent nano robotics [7].

DNA has been considered to be the most promising molecules for building 2D and 3D nanostructures such as DNA tile and DNA box [8-12]. In addition, a variety of DNA nanomachines were built that can perform mechanical functions such as scission, directional motion and rolling powered by molecular fuels [13-17]. All these interesting nanodevices can be designed directly using a computer program because DNA has the simple recognition rule as forming only stable A-T and G-C Watson-Crick base pair. The presence of one or more mismatched base pairs decreases greatly the stability of corresponding duplex. However, their practical applications remain a challenge, especially for biological purposes. On the other hand, functional molecules such as fluorophores, dyes, ligands, and photoresponsive molecules have been introduced into DNA for attaining more functionality for biological applications [18-21]. If these functional molecules can be introduced into DNA nanostructures or used mainly as the nanomaterials themselves, their applications should be widely extended.

Recently multiple azobenzene residues have been introduced to DNA for either photoregulation of biological functions or construction of light-driven nanodevices [22-25]. Several basic motifs such as homo-cluster, hetero-cluster, and interstrand-wedged duplex have been constructed [26-30]. However, the number of azobenzenes introduced was never more than that of base pairs. In this study, we designed a nanomaterial involving much more azobenzenes than base pairs. Rather than as a modified DNA, the novel molecule prefers to be looked as an artificial macromolecule with side chain containing azobenzenes as photoresponsive molecules and nucleobases providing recognition ability. It can also be classified as one of the foldamers defined loosely as "polymers with a strong tendency to adopt a specific compact conformation" [5]. The synthesized foldamer can form a stable duplex with high recognition ability and excellent photoresponsiveness so that the construction of highly efficient light-driven nanodevice becomes possible.

2 Results and Discussion

Formation of the artificial duplex involving azobenzene pairs and base pairs from two complementary foldamer chains. As illustrated in Fig. 1a, azobenzene (Azo), one of the photoresponsive molecules that can carry out reversible *trans-cis* photoisomerization, was used as a model functional molecule to synthesize a new foldamer. Each of the azobenzene was attached to a D-threoninol linker for being inserted into the foldamer chain with standard phosphoramidite chemistry. Accordingly, as shown in Fig. 1a, the azobenzene residue (X) and the nucleotide (A, T, G, or C) could be polymerized in a sequence-controlled way. Some of the sequences used in this study were shown in Fig. 1b. In foldamer A7X and B7X, for example, seven azobenzene residues (Xs) and 10 nucleotides were involved. When a duplex formed between A7X and B7X, an artificial duplex involving 14 azobenzenes and 10 base pairs should be formed as shown in Fig. 1a. As an azobenzene molecule has the similar size as a base pair, each two azobenzene moieties from complementary strands will stack and overlap with each other to form an azobenzene-azobenzene (Azo-Azo) pair. In the middle part of A7X/B7X duplex, each base pair is sandwiched between four azobenzenes or two Azo-Azo pairs. Obviously, the pairing of Azo to Azo is different from the base-pairing through hydrogen bond. More interestingly, when the azobenzene moieties are isomerized to the non-planar *cis* form, the duplex is expected to be completely dissociated due to the serious steric hindrance of so many *cis*-azobenzenes so that the complete photoregulation becomes possible.



Name	Sequence
A-n	5'-CGTTAGTTCA-3'
B-n	3'-GCAATCAAGT-5'
A7X	5'-CGXTXTXAXGXTXTXCA-3'
B7X	3'-GCXAXAXTXCXAXAXGT-5'
B7X-C	3'-GCXAXAXCXCXAXAXGT-5'
A3X	5'-CG TXT AXG TXT CA-3'
B4X	3'-GCXA AXT CXA AXGT-5'

Fig. 1. Schematic illustration of the artificial duplex with Azo-Azo pairing motif formed between two foldamer chains (a) and some sequences of the foldamers used in this study (b). The complete photoswiching of the duplex formation becomes possible due to the presence of so many azobenzene moieties that are even more than base pairs.

The stability of these newly constructed duplexes was evaluated by melting temperature (T_m) measurement. The T_m curves of duplexes A7X/B7X, A7X/B7X-C (containing an A-C mismatched base pair), and A7X/B4X as well as single-stranded B7X are shown in Fig. 2a. A typical sigmoid T_m curve was obtained for A7X/B7X, and the T_m value was as high as 55.9°C, which is 25.4°C higher than that of the natural duplex A-n/B-n with the same nucleotide sequence (Fig. 2b). Here, the T_m for A7X/B7X was measured at 375 nm, at which the spectra of introduced azobenzenes changed greatly with duplex formation. Similar results were also obtained when T_m was measured at

295 nm and 335 nm (data not shown). When B4X involving only 4 azobenzenes was used instead of B7X, the T_m of A7X/B4X decreased greatly to 37.4 °C, which was even lower than that of A7X/B7X-C. However, the T_m of A3X/B4X (designated as the interstrand-wedged motif) was as high as 55.0°C, indicating that the symmetry is important for forming stable duplex. For the similar reason, A7X could hardly form duplex with native DNA B-n and T_m of A7X/B-n was estimated to be below 10°C (Fig. 2b). As a result, A7X recognized B7X but not B4X and B-n, while B4X recognized A3X but not A7X, although they have the same nucleotide sequences. For the duplex A7X/B7X-C, in which an A-C mismatch was introduced, the T_m decreased by 13.4 °C to 42.5°C, demonstrating that the newly constructed duplex has high recognition ability, which is comparable to the native DNA duplex. In addition, no slowness of the hybridization dynamics was observed during T_m measurement. Thus, a novel artificial duplex involving Azo-Azo pairs and base pairs with both high stability and specificity was constructed.

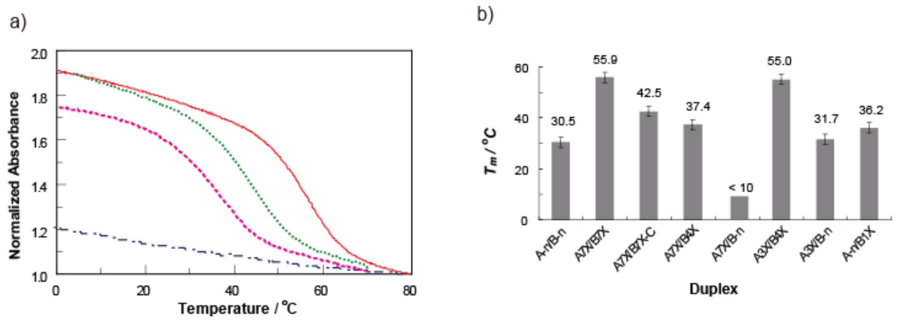


Fig. 2. (a) Melting curves of duplex A7X/B7X (solid line), A7X/B7X-C (dotted line), and A7X/B4X (dashed line). The result of single-stranded B7X is also shown. The absorbance change with temperatures was recorded at 375 nm. The T_m values are shown in (b). Conditions: 2.0 μM oligo, pH7.0 (10 mM Na_2HPO_4), 100 mM NaCl.

Usually, the T_m of a native DNA duplex is measured by recording the hypochromic effect due to duplex formation at 260 nm that is the maximum absorption of nucleobases. Here, the T_m was measured by monitoring the absorbance change at 375 nm (where azobenzenes but not nucleobases have absorption) with temperatures [27-30]. As show in Fig. 3a, the absorbance at 260 nm changed little with temperatures for A7X/B7X, probably because the hypochromic effect of duplex formation was counteracted by the increase of absorbance caused by the spectra shift of azobenzene at 230-280 nm. However, both of the absorbance and the spectrum shape of π-π* transition of azobenzene changed greatly with temperatures. These changes are similar with those reported previously, which were caused by forming one Azo-Azo pair (H-type dimer) within a DNA duplex [28]. Thus, the spectra changed here should come from the hybridization of A7X with B7X. As shown in Fig. 3b, when only one strand such as B7X was present, the intramolecular interaction only caused little red shift of the spectrum.

Structure analysis of the duplex involving Azo-Azo pairing motif. As reported previously, when an Azo-Azo pair forms within a DNA duplex (5′-GGTATCXGCAATC-3′/3′-CCATAGXCGTTAG-5′), the two azobenzene moieties stack with each other to form an H-aggregation [28]. Here, the similar spectrum change was observed, indicating that Azo-Azo pairs formed to give H-like aggregations (Fig. 3a). For structural analysis, the circular dichroism (CD) of duplex A7X/B7X was also measured (Fig. 3c). At temperatures below its T_m, a strong and symmetric positive-negative Cotton effect was induced at around 335 nm, which is the λ_{max} of azobenzene (π-π^* transition). On the other hand, when the duplex dissociated at 80°C, only a weak negative CD due to the single-stranded A7X and B7X was observed. The similar negative CD spectra were also obtained when only single-stranded B7X was present (Fig 3d). Even at 0°C and 10 μM of B7X was used in the absence of A7X, only the strength became stronger and the shape did not change, indicating that no intermolecular interaction occurred (data not shown). At 260 nm, very weak or only negative Cotton effect was observed, probably because the CD signals belonging to azobenzene at 220-280 nm are negative. Most of the base pairs are separated by two azobenzene moieties might be another reason for the weak Cotton effect at 260 nm. All the results showed that a right-handed double helix formed between A7X and B7X.

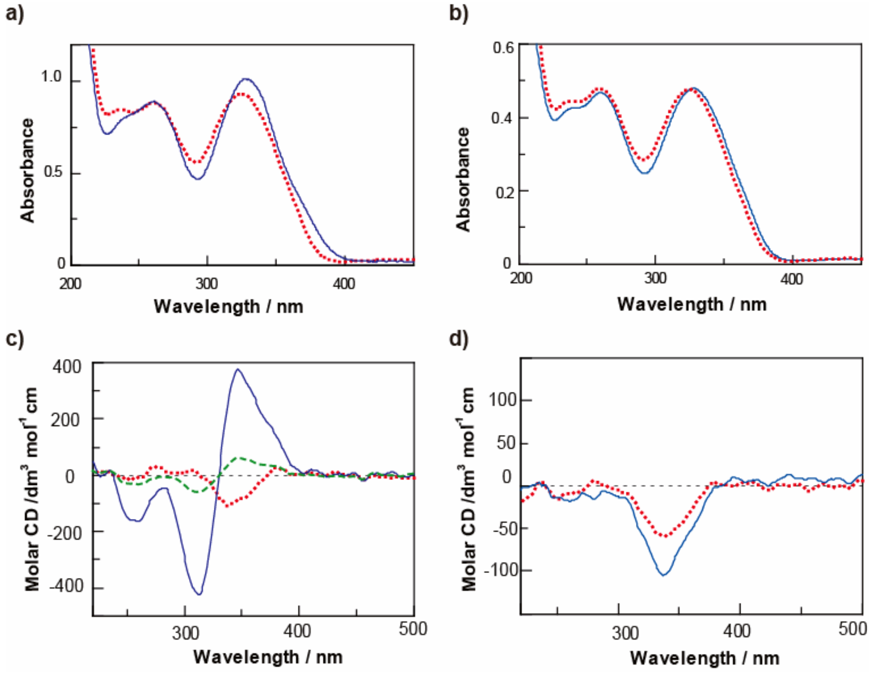


Fig. 3. UV/Vis (a and b) and CD (c and d) spectra of A7X/B7X (a and c) and B7X (b and d) at 80°C (dotted lines) and 20°C (solid lines). The CD spectrum at 0°C after UV light irradiation (dashed line) is also shown in (c). Conditions: 2.0 μM oligo, pH7.0 (10 mM Na_2HPO_4), 100 mM NaCl.

The molecular modeling structure of A7X/B7X is shown in Fig. 4c. It can be clearly seen that a regular right-handed duplex with seven Azo-Azo pairs and 10 base pairs forms. All the azobenzene moieties protrude slightly in the major groove, which is much wider than that of natural B-form DNA duplex. Each Azo-Azo pair consists of two azobenzene moieties stacking well with each other. Each base pair also stacks well with two adjacent azobenzene moieties. For the obtained structure, it can be estimated that one pitch of the duplex consists of about 7 base pairs and 7 Azo-Azo pairs (or 14 azobenzene moieties). It can also be seen that the two azobenzene moieties are almost antiparallel to each other, which is consistent with the NMR structure obtained when an Azo-Methyl Red pair is introduced into a 6-bp-long DNA duplex [28]. However, two adjacent Azo-Azo pairs which are separated by a base pair wind forward in a right-handed way. The strong positive-negative Cotton effect in the CD spectrum may be a total effect of the winding. Interestingly when only one Azo-Azo pair was present, the CD showed a weak negative-positive Cotton effect, indicating that the two azobenzene moieties stack together with a slight left-hand winding (data not shown). As compared with the native DNA duplex, the phosphodiester backbone in A7X/B7X is lengthened to some extent, and the distance between each two phosphates becomes longer. For comparison, the modeling structure of interstrand-wedged motif is also shown [29,30], which results in a similar CD spectrum as duplex A7X/B7X (described later in detail).

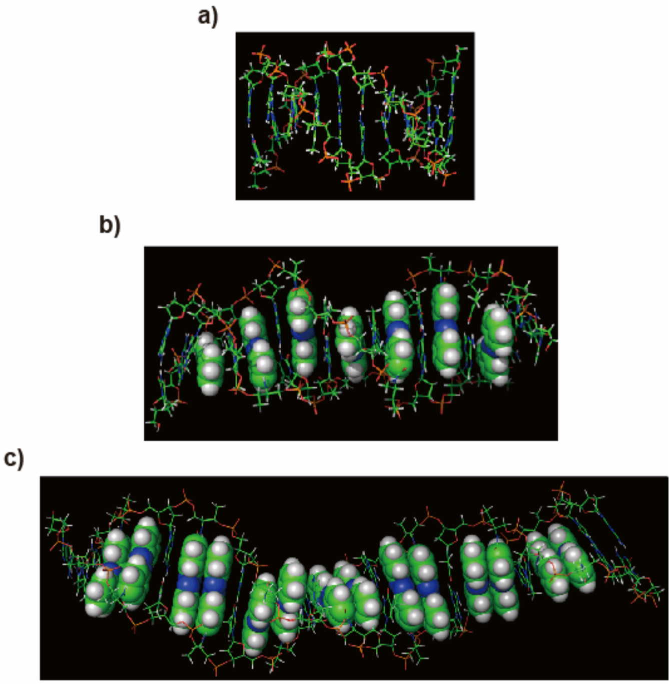


Fig. 4. Molecular modeling structures of duplex A-n/B-n (a), A3X/B4X (b), and A7X/B7X (c). The azobenzene moieties are highlighted in a space-filling (CPK) model.

Photoreponsiveness of duplex A7X/B7X. Here, the azobenzenzene that has good photoresponsiveness was used as a model functional molecule to build the foldamer. As reported previously, the DNA hybridization has been reversibly photoregulated by introducing multiple azobenzenes into DNA [22, 29, 30]. However, the efficiency of *cis-to-trans* photoisomerization became much lower at a temperature below 37°C, when a stable duplex was formed. Accordingly, more introduced azobenzenes make the complete ON-OFF photoregulation easier. In the case of A7X/B7X, for example, 5 azobenzenes should be in *cis*-form and enough to cause complete dissociation of the duplex even when 30% *cis-to-trans* photoisomerization occurs. Interestingly, as shown in Fig. 5, the photoisomerization did not decrease greatly as compared with the single-stranded state. Even when the photoirradiation was carried out at 25°C, about 40% of all the azobenzenes were isomerized to *cis* form. At 37°C, the similar results of photoisomerization were obtained for duplex A7X/7BX and single-stranded B7X, demonstrating that the duplex was almost completely dissociated at 37°C (Fig. 5). As shown in Fig. 3c, very weak CD was obtained for *cis*-A7X/B7X even at 0°C. From the T_m curve shown in Fig. 2a, almost all the duplexes formed at 37°C for *trans* form. The T_m of *cis* form was very low and could not be measured by monitoring the absorbance change with temperature (data not shown). As a result, the complete ON-OFF photoregulation of artificial duplex was attained.

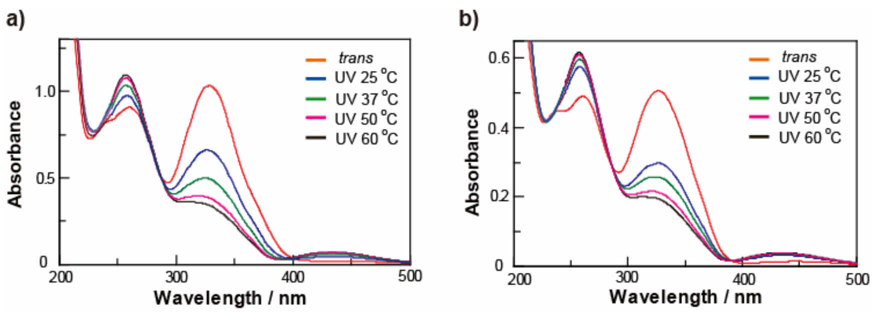


Fig. 5. *cis-to-trans* photoisomerization of azobenzenes in duplex A7X/B7X (a) and single-stranded B7X (b) at various temperatures. The photoregulation was carried out at 25, 37, 50, and 60°C, respectively. Spectra are measured at 25°C.

Comparison of Azo-Azo pairing duplex with interstrand-wedged duplex. Recently we have reported that an interstrand-wedged duplex such as A3X/B4X was constructed [29,30]. As shown in Fig. 4b, azobenzene moieties and base pairs are lined up alternatively, and the ratio of azobenzene and base pairs is 1:1. In this interstrand-wedged duplex, azobenzenes only stack with base pair but not another azobenzene. As shown in Fig. 6a, an obvious red shift with the duplex formation was observed due to the stacking of azobenzene with base pairs. Certainly, the spectra did not change in the way of forming H-aggregation (See Fig. 2a for comparison).

Although the two motifs have quite different structures, their CD spectra are very similar (Fig. 6b). The molar CD of duplex A7X/B7X is almost doubled as compared with that of A3X/A4X, because A7X/B7X has 14 azobenzene moieties but A3X/A4X only has 7 ones. The two motifs also have similar sequence specificity for hybridization.

Fig. 7 shows the T_ms of all the 16 combinations including four duplexes full match and 12 mismatched ones. Most of mismatched duplexes have a T_m more than 10°C lower than that of full match ones. In addition, both motifs showed high photoresponsiveness which makes possible the complete ON-OFF photoregulation of the duplex formation. These results demonstrated again that azobenzene moiety has an excellent compatibility with the base pair, and so does the D-threoninol with D-ribose in the phosphodiester backbone. Both motifs showed the good properties as nanomaterials for constructing nanostructures and nanodevices.

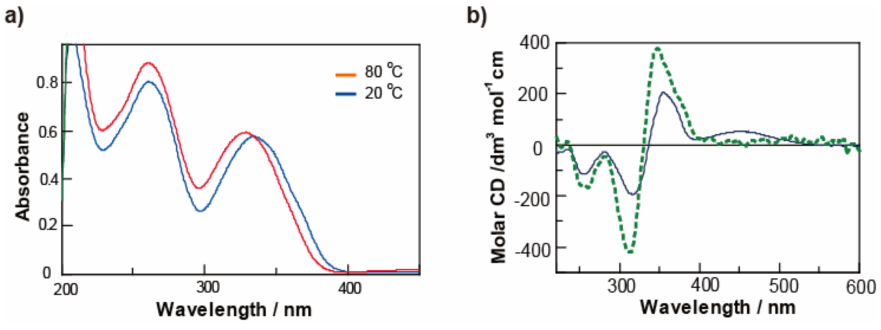


Fig. 6. UV/Vis (a) and CD (b) spectra of A3X/B4X (solid line) with interstrand-wedged motif. CD spectra are measured at 20°C. For comparison, the CD spectrum for duplex A7X/B7X is shown (dashed line). Conditions: 2.0 μM oligo, pH7.0 (10 mM Na_2HPO_4), 100 mM NaCl.

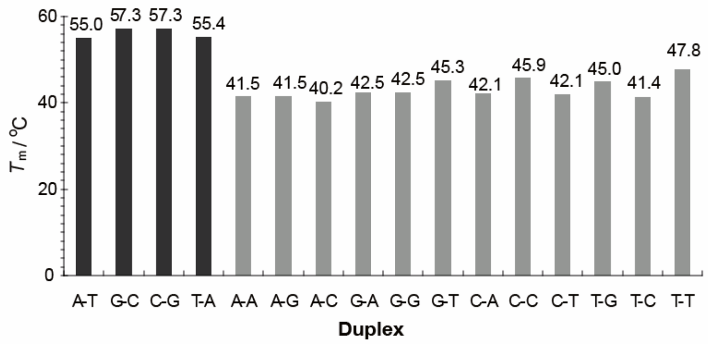


Fig. 7. T_m of duplexes 5′-CGTXTAXMTXTCA-3′/3′-GCXAAXTNXAAXGT-3′ (M, N = A, C, G or T). For example, duplex **A-T** indicates that M = A and N = T. Conditions: 4.0 μM oligo, pH7.0 (10 mM Na_2HPO_4), 100 mM NaCl.

3 Conclusions

In conclusion, a novel nanomaterial composed of functional molecules and base pairs was constructed. For the Azo-Azo pairing motif, a stable duplex with high specificity formed even when the ratio of azobenzene moieties and base pairs were 2:1. The strong stacking effect between hydrophobic azobenzenes and that between azobenzene and base pair account for the high stability. On the other hand, the base pairing provides the sequence specificity. Both the strong positive-negative Cotton effect of CD and the modeling structure revealed that the Azo-Azo pairing motif is a regular right-handed duplex. Furthermore, for both Azo-Azo pairing and interstrand-wedged motif, the duplex formation can be completely switched by light irradiation.

As azobenzenes tethered on D-threoninol and nucleosides on D-ribose can be arranged with any sequence by using standard phosphoramidite chemistry, any sequence can be synthesized [22]. Other functional molecules with the similar size can also be introduced with this approach so that it can be used as a general strategy for designing artificial nanomaterials. With the combination of clustering motif and the motifs presented here, stable duplex involving functional molecules and base pairs with any ratio can be constructed. Consequently, a variety of functional nanostructures and nanodevices are promising to be built. The high hybridization specificity of these new motifs can be used to form sticky ends as new glues for constructing nanostructures. When the functional molecules are fluorophores and quenchers, for example, a supra-molecular beacon with high sensitivity for nucleic detection may be constructed. Construction of these motifs with other functional molecules and the design of functional nanostructures such as light-driven nanodevices are underway [25].

4 Experimental Section

Materials. The oligonucleotides consisting of only natural bases were supplied by Integrated DNA Technologies, Inc. (Coralville, USA). The oligonucleotides involving azobenzene residues were supplied by Nihon Techno Service Co., Ltd. (Tsukuba, Japan), and purified by HPLC. Concentrations of oligonucleotides were determined by UV-Vis spectroscopy analysis within an error margin of 10%. The molecular extinction coefficient (ε) of an azobenzene residue at 260 nm is 7.0×10^3 mol L^{-1} cm^{-1}.

Measurement of Melting Temperatures. Melting curves were measured by monitoring the absorbance change at a certain wavelength (260, 295, 360, or 375 nm) with temperature (1.0 °C min^{-1}) using a JASCO model V-530 spectrometer equipped with a programmable temperature-controller (JASCO, Tokyo, Japan) or a Shimadzu UV spectrophotometer UV-1800 (Shimadzu, Kyoto, Japan). The melting temperature (T_m) was determined from the maximum of the first derivative of each melting curve.

UV/Vis Spectroscopy. UV-Vis spectra of samples in a buffer containing 100 mM NaCl and 10 mM NaH_2PO_4 (pH 7.0) were measured using a JASCO model V-530 spectrometer (JASCO, Tokyo, Japan). A cuvette with an optical path length of either 1 or 10 mm was used. The solved air was purged by heating the DNA solution to 90°C and maintaining it at 90°C for longer than 3 min.

CD Spectra Measurement. CD spectra (0.3 or 2.0 mL) in a phosphate buffer (100 mM NaCl, pH 7.0) were measured using a JASCO J-820 CD spectropolarimeter equipped with a programmed temperature-controller (JASCO, Tokyo, Japan). A cuvette with an optical path length of either 1 or 10 mm was used.

Molecular Modeling. The Insight II/Discover 98.0 program package (Accelrys Software Inc., San Diego, USA) was used for molecular modeling to obtain energy-minimized structures by minimization of the conformation energy [29,30]. The effects of water and counterions were simulated by a sigmoidal, distance-dependent, dielectric function. The B-type duplex was used as the initial structure, and the AMBER force field was used for calculation. Structures of azobenzene residues were built using the attached graphical program.

Acknowledgments

This work was supported by Core Research for Evolution Science and Technology (CREST), Japan Science and Technology Agency (JST). Partial support was provided by a Grant-in-Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Seeman, N.C., Lukeman, P.S.: Nucleic Acid Nanostructures: Bottom-Up Control of Geometry on the Nanoscale. Rep. Prog. Phys. 68, 237–270 (2005)
2. Fujita, M., Tominaga, M., Hori, A., Therrien, B.: Coordination Assemblies from a Pd(II)-Cornered Square Complex. Acc. Chem. Res. 38, 369–378 (2005)
3. Zheng, J.P., Birktoft, J.J., Chen, Y., Wang, T., Sha, R.J., Constantinou, P.E., Ginell, S.L., Mao, C.D., Seeman, N.C.: From Molecular to Macroscopic via the Rational Design of a Self-Assembled 3D DNA Crystal. Nature 461, 74–77 (2001)
4. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and Autonomous Computing Machine Made of Biomolecules. Nature 414, 430–434 (2001)
5. Hecht, S., Huc, I.: Foldamers: Structure, Properties, and Applications. Wiley-VCH Verlag GmBH & Co. KGaA, Weinheim (2007)
6. Kay, E.R., Leigh, D.A., Zerbetto, F.: Synthetic Molecular Motors and Mechanical Machines. Angew. Chem., Int. Ed. 46, 72–191 (2007)
7. Hamdi, M., Ferreira, A.: DNA nanorobotics. Microelectronics J. 39, 1051–1059 (2008)
8. Shih, W.M., Quispe, J.D., Joyce, G.F.: A 1.7-Kilobase Single-Stranded DNA that Folds into a Nanoscale Octahedron. Nature 427, 618–621 (2004)
9. Mirkin, C.A., Letsinger, R.L., Mucic, R.C., Storhoff, J.J.: A DNA-Based Method for Rationally Assembling Nanoparticles into Macroscopic Materials. Nature 382, 607–609 (1996)
10. Sharma, J., Chhabra, R., Cheng, A., Brownell, J., Liu, Y., Yan, H.: Control of Self-Assembly of DNA Tubules Through Integration of Gold Nanoparticles. Science 323, 112–116 (2009)
11. He, Y., Ye, T., Su, M., Zhang, C., Ribbe, A.E., Jiang, W., Mao, C.D.: Hierarchical self-assembly of DNA into symmetric supramolecular polyhedra. Nature 452, 198–201 (2008)
12. Andersen, E.S., Dong, M., Nielsen, M.M., Jahn, K., Subramani, R., Mamdouh, W., Golas, M.M., Sander, B., Stark, H., Oliveira, C.L.P., Pedersen, J.S., Birkedal, V., Besenbacher, F., Gothelf, K.V., Kjems, J.: Self-assembly of a nanoscale DNA box with a controllable lid. Nature 459, 73–77 (2009)
13. Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: A DNA-Fuelled Molecular Machine Made of DNA. Nature 406, 605–608 (2000)

14. Shin, J.S., Pierce, N.A.: A Synthetic DNA Walker for Molecular Transport. J. Am. Chem. Soc. 126, 10834–10835 (2004)
15. Seeman, N.C.: From Genes to Machines: DNA Nanomechanical Devices. Trends. Biochem. Sci. 30, 119–125 (2005)
16. Beissenhirtz, M.K., Willner, I.: DNA-Based Machines. Org. Biomol. Chem. 4, 3392–3401 (2006)
17. Beyer, S., Simmel, F.C.: A Modular DNA Signal Translator for the Controlled Release of a Protein by an Aptamer. Nucleic Acid Res. 34, 1581–1587 (2006)
18. Kutyavin, I.V., Afonina, I.A., Mills, A., Gorn, V.V., Lukhtanov, E.A., Belousov, E.S., Singer, M.J., Walburger, D.K., Lokhov, S.G., Gall, A.A., Dempcy, R., Reed, M.W., Meyer, R.B., Hedgpeth, J.: 3'-minor groove binder-DNA probes increase sequence specificity at PCR extension temperatures. Nucleic Acids Res. 28, 655–661 (2000)
19. Wang, K., Tang, Z., Yang, C.J., Kim, Y., Fang, X., Li, W., Wu, Y., Medley, C.D., Cao, Z., Li, J., Colon, P., Lin, H., Tan, W.: Molecular engineering of DNA: molecular beacons. Angew. Chem. Int. Ed. 47, 2–17 (2008)
20. Kelley, S.O., Boon, E.M., Barton, J.K., Jackson, N.M., Hill, M.G.: Single-base mismatch detection based on charge transduction through DNA. Nucleic Acids Res. 27, 4830–4837 (2000)
21. Mayer, G., Heckel, A.: Biologically active molecules with a "light switch". Angew. Chem. Int. Ed. Eng. 45, 4900–4921 (2006)
22. Asanuma, H., Liang, X.G., Nishioka, H., Matsunaga, D., Liu, M.Z., Komiyama, M.: Synthesis of Azobenzene-Tethered DNA for Reversible Photo-Regulation of DNA Functions: Hybridization and Transcription. Nat. Protocols 2, 203–212 (2007)
23. Liang, X.G., Nishioka, H., Takenaka, N., Asanuma, H.: A DNA Nanomachine Powered by Light Irradiation. ChemBioChem. 9, 702–705 (2008)
24. Liang, X.G., Nishioka, H., Takenaka, N., Asanuma, H.: Construction of Photon-Fueled DNA Nanomachines by Tethering Azobenzenes as Engines. In: Goel, A., Simmel, F.C., Sosík, P. (eds.) DNA 14. LNCS, vol. 5347, pp. 21–32. Springer, Heidelberg (2009)
25. Zhou, M.G., Liang, X.G., Mochizuki, T., Asanuma, H.: A light-driven DNA nanomachine for efficiently photoswitching RNA digestion. Angew. Chem. Int. Ed. 49, 2167–2170 (2010)
26. Asanuma, H., Shirasuka, K., Takarada, T., Kashida, H., Komiyama, M.: DNA-Dye Conjugates for Controllable H* Aggregation. J. Am. Chem. Soc. 125, 2217–2223 (2003)
27. Kashida, H., Fujii, T., Asanuma, H.: Threoninol as a Scaffold of Dyes (Threoninol-nucleotide) and Their Stable Interstrand Clustering in Duplexes. Org. Biomol. Chem. 6, 2892–2899 (2008)
28. Fujii, T., Kashida, H., Asanuma, H.: Analysis of Coherent Heteroclustering of Different Dyes by Use of Threoninol-Nucleotides for Comparison with the Molecular Exciton Theory. Chem. Eur. J. 15, 10092–10102 (2009)
29. Liang, X.G., Mochizuki, T., Asanuma, H.: A Supra-Photoswitch Involving Sandwiched DNA Base Pairs and Azobenzenes for Light-Driven Nanostructures and Nanodevices. Small 5, 1761–1768 (2009)
30. Liang, X.G., Nishioka, H., Mochizuki, T., Asanuma, H.: An interstrand-wedged duplex composed of alternating DNA base pairs and covalently attached intercalators. J. Mater. Chem. 20, 575–581 (2010)

Efficient Turing-Universal Computation with DNA Polymers

Lulu Qian[1], David Soloveichik[4], and Erik Winfree[1,2,3]

[1] Bioengineering, California Institute of Technology,
Pasadena, CA 91125, USA
luluqian@caltech.edu
[2] Computer Science
[3] Computation & Neural Systems, California Institute of Technology,
Pasadena, CA 91125, USA
winfree@caltech.edu
[4] Computer Science & Engineering, University of Washington,
Seattle, WA 98195, USA
dsolov@u.washington.edu

Abstract. Bennett's proposed chemical Turing machine is one of the most important thought experiments in the study of the thermodynamics of computation. Yet the sophistication of molecular engineering required to physically construct Bennett's hypothetical polymer substrate and enzymes has deterred experimental implementations. Here we propose a chemical implementation of stack machines — a Turing-universal model of computation similar to Turing machines — using DNA strand displacement cascades as the underlying chemical primitive. More specifically, the mechanism described herein is the addition and removal of monomers from the end of a DNA polymer, controlled by strand displacement logic. We capture the motivating feature of Bennett's scheme: that physical reversibility corresponds to logically reversible computation, and arbitrarily little energy per computation step is required. Further, as a method of embedding logic control into chemical and biological systems, polymer-based chemical computation is significantly more efficient than geometry-free chemical reaction networks.

1 Introduction

With the birth of molecular biology 70 years ago came the realization that the processes within biological cells are carried out by molecular machines, and that the most central processes involved the manipulation of information-bearing polymers. Roughly 30 years ago, Charles Bennett took that vision one step further by recognizing that arbitrarily complex information processing could be carried out, in principle, by molecular machines of no greater complexity than those already observed in nature [5,6]. Based on the intrinsic reversibility of chemical reactions, Bennett used this insight to give a thermodynamic argument that there is no fundamental energetic cost to computation — only a cost to erase data. This conclusion derives from four principles: (1) as Landauer observed [15], making a logically irreversible decision entails an energetic expenditure of $kT \ln 2$, and thus there is an unavoidable cost to irreversible logical operations; (2) being logically reversible is not enough to ensure low-energy computation, since it is possible to implement reversible logic using irreversible mechanisms; (3) a physically reversible system with an essentially linear state space can be biased ever-so-slightly forward, in which case progress is made despite involving a Brownian random walk, with the mean speed being linear in the (arbitrarily near zero) energy expended per step; and (4) any logically irreversible computation can be recast with a minimal number of extra computational steps [5,7] as a logically reversible computation that requires irreversible operations only when preparing input and output during repeated use. It's intriguing to ask whether Landauer's and Bennett's principles have any bearing on the remarkable efficiency of living things, but cellular processes typically use several times more energy than needed for logical irreversibility. On the other hand, modern electrical computers expend many orders of magnitude more energy than required by logical irreversibility, presenting the challenge of building computers that have the efficiency Bennett argued is possible.

Direct implementation of Bennett's hypothetical chemical Turing machine has been hampered by our inability, as yet, to engineer molecular machinery to spec. Len Adleman's laboratory demonstration of a DNA computing paradigm for solving NP-complete problems [1] ignited renewed interest in the molecular implementation of Turing machines. Early theoretical proposals made use of existing enzymes but required a series of laboratory manipulations to step the molecular Turing machines through their operational cycle [20,2,23], while later theoretical proposals suggested how autonomous molecular Turing machines could be built but made use of hypothetical enzymes or DNA nanostructures [14,4,11,28,12]. Experimental demonstrations of autonomous biomolecular computers implemented weaker models of computation such as digital circuits or finite state machines [22,3]. Two-dimensional molecular self-assembly is Turing universal [26], implementable with DNA tiles [21], and can be physically and logically reversible [27], but it has the distinct disadvantage of storing the entire history of its computation within a supramolecular complex — it's bulky.

Recent work has pointed to an alternative to geometrical organization (in polymers or crystals) as the basis for Turing-universal molecular computation: abstract chemical reaction networks (CRNs) with a finite number of species in a well-mixed solution are structurally simple enough (essentially geometry-free) that in principle arbitrary networks can be implemented with DNA [25], yet they are (probabilistically) Turing universal [24]. This Turing universal computation using geometry-free chemical reaction networks is theoretically accurate and reasonably fast (only a polynomial slowdown), but requires molecular counts (and therefore volumes) that grow exponentially with the amount of memory used [16,24]. In contrast, reaction networks using heterogeneous polymers — the simplest kind of geometrical organization, as in Bennett's vision — can store all information as strings within a single polymer, therefore requiring volume that grows only linearly with the memory usage [6,14]. Further, geometry-free models are not energy efficient, requiring much more than Landauer's energy limit because the computation must be driven irreversibly forward to avoid error. Here, we combine the advances in geometry-free CRN implementation [25] with a simple DNA polymer reaction primitive to obtain a plausible DNA implementation of time- and space- and energy-efficient Turing-universal computation. Our construction requires a small fixed number of polymers, thereby having the same efficient linear memory/volume tradeoff as Bennett's hypothetical scheme; it also has a time complexity nearly as good (only a quadratic slowdown). As in Bennett's scheme, the time complexity scales linearly with energy use (for small energies). Both constructions can perform irreversible computation using the minimum achievable amount of energy per step, $kT \ln m$, where m is the mean number of immediate predecessors to the logical states of the Turing machine simulation. (This energy bound is 0 for reversible Turing machines).

Our constructions will consist of two parts. First, a geometry-free chemical reaction network, and secondly, reactions involved in polymer modification. While the polymer modification reactions will perform the essential job of information storage and retrieval, the geometry-free reaction network will perform the logic operations. We describe the necessary elements for the implementation of relevant geometry-free chemical reaction networks in section 2. In the following section 3 we describe the polymer reactions. Based on these two DNA implementation schemes, section 4 shows how they can be used to efficiently simulate stack machines. Finally, in section 5 we show how Bennett's logically reversible Turing machines can be implemented with physically reversible DNA reactions. We conclude by evaluating our contributions and pointing out room for further improvement.

2 Irreversible and Reversible Chemical Reaction Networks

In this section we discuss the components necessary for the implementation of the geometry-free chemical reaction network part of our constructions.

Recent work has proposed a DNA implementation of arbitrary (geometry-free) chemical reaction networks [25], with which we assume the reader is familiar. (Even so, the construction given here is self-contained.) However, thermodynamic reversibility was not considered. Indeed, reversible reactions would simply correspond to two separate forward and reverse reactions, with both reactions having to be independently driven irreversibly by chemical potential energy provided by DNA fuels. This is wasteful for the consumption of both energy and fuel reagents. The construction we develop in this section is entirely physically reversible in the sense that firing a sequence of forward reactions and then the reverse sequence of the corresponding reverse reactions brings the chemical system into the same exact physical state as it was in the beginning, including recovery of fuel reagents and any energy used.

The major challenge in adapting the scheme of ref. [25] to implement reversible chemical reactions in a physically reversible manner, is that the exact DNA strand representing a particular signal species is a function of not just the signal, but also

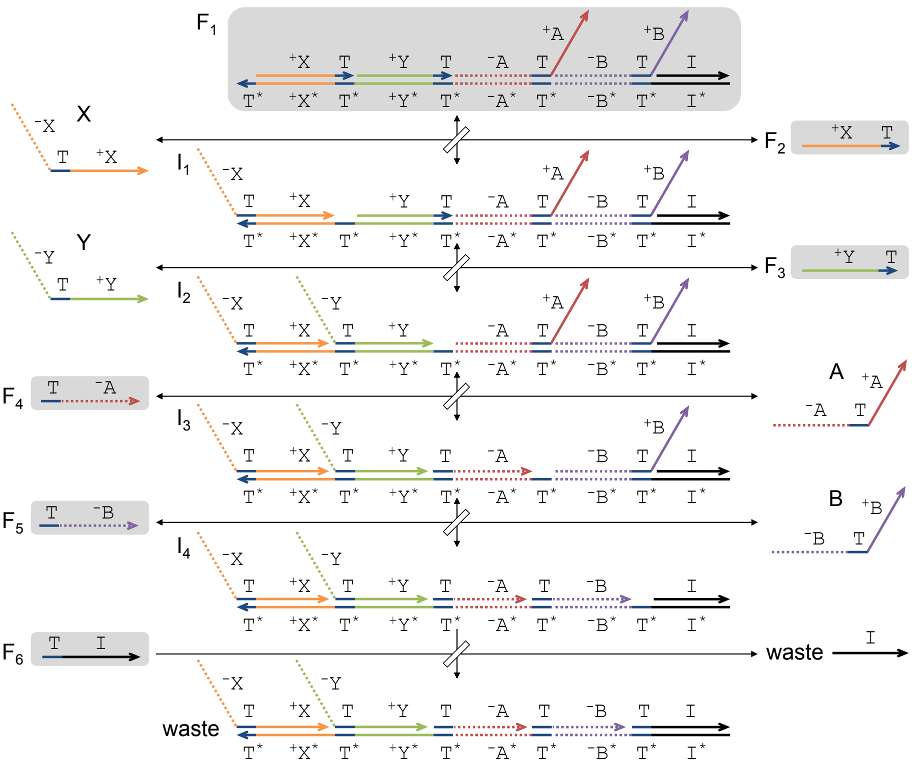


Fig. 1. The implementation of the formal bimolecular reaction $X + Y \rightarrow A + B$ using history-free signal species X, Y, A, B. Each strand displacement reaction is shown, with arrowed thin black lines connected by diagonal rectangles indicating reactants, products and reversibility. F_1 through F_6 are fuel species mediating the formal reaction, and are present in high concentration. F_1 is unique to this reaction, while F_2 through F_6 may be shared with other reactions. I_1 through I_4 are intermediates of the formal reaction and they are all unique to this reaction. Domains ^+X, ^-X, ^+Y, ^-Y, ^+A, ^-A, ^+B, ^-B are specific to formal species X, Y, A, B respectively, while domain I is used for the irreversible step in all reactions, and the short toehold domain T is used universally wherever a toehold is needed. The asterisk indicates Watson-Crick complementary domains, e.g. $^+Y^*$ is complementary to ^+Y.

of the formal reaction that produced it. Specifically, all signal strands contain a "history" domain that holds the strand in an inactive form before release from a DNA fuel complex. So even if we were to make every strand displacement step reversible, the reverse reaction of $X \rightarrow Y$ would only be able to uptake signal species Y that have the correct history domains for this reaction, rather than the entire population of Y which may have been generated by other reactions.

In order to solve this problem, we develop a "history-free" implementation of arbitrary chemical reaction networks. (Cardelli has proposed an elegant and even further reduced scheme [10] that is also history-free, but it appears unsuitable for our polymer reaction construction.) We describe an irreversible scheme, that with slight modifications can become reversible. In addition to making the reversible scheme straightforward, the history-free signal strand motif simplifies the correspondence between the abstract CRN and the DNA implementation: each CRN species now corresponds to exactly one DNA species.

Like ref. [9], but unlike ref. [25], we use stochastic semantics in this paper where reactions manipulate integer molecular counts of the reacting species, rather than real-valued concentrations. The applicable kinetic and thermodynamic laws are widely known from the consideration of small-scale chemical systems. Unlike the quantitative kinetics requirements of ref. [25], in the context of this paper a successful implementation of an irreversible reaction such as $X + Y \rightarrow A + B$ must simply be qualitatively correct by satisfying two conditions: First, there must be some overall irreversible reaction pathway that first consumes a molecule of X, a molecule of Y, and then produces a molecule of A and a molecule of B. Second, the reaction pathway must become irreversible at some point only *after* X and Y have been consumed. If the pathway were to become irreversible before Y is consumed, then in the absence of Y, X would still be used up. An implementation of a reversible reaction $X + Y \rightleftharpoons A + B$ must be a reversible reaction pathway that first consumes a molecule of X, a molecule of Y, and then produces a molecule of A and a molecule of B. While we do not explicitly address the question of quantitatively correct reaction kinetics, our constructions respect the usual scaling laws for kinetics of unimolecular, bimolecular, and higher-order reactions; similar techniques as in ref. [25] could applied to these constructions.

Fig. 1 shows the history-free implementation of the irreversible reaction $X + Y \rightarrow A + B$, and Fig. 2 shows the corresponding implementation of the reversible reaction $X + Y \rightleftharpoons A + B$.

In Fig. 1, fuel DNA species F_1, F_2, F_3, F_4, F_5 and F_6 are initially present in high concentration, and we assume they remain present in high concentration throughout. Signal DNA species X, Y, A and B are present in low amounts relative to the fuel species and indicate meaningful signals. To make the reaction module composable, all signal DNA species are of the same form, and allow the coupling of such formal reactions together. (They are also of the same form as signal species in another DNA strand displacement network architecture [19], which allows even broader couplings.) All signal species have one short toehold domain in the middle, one long recognition domain "$-$" on the $5'$ end and another long recognition domain "$+$" on the $3'$ end. The bottom strand of fuel F_1 has five long recognition domains connected by five short toehold domains. Initially, the left-most toehold domain is single-stranded and is thus available for binding, and the other four toeholds are double-stranded and thus sequestered.

Signal X first binds to fuel F_1 by the exposed toehold and branch migration occurs through domain ^{+}X. The top strand F_2 will fall off when it's only held to the bottom strand by the toehold and leave F_1 as intermediate product I_1. (This is the principle of toehold exchange [29,30].) Compared to F_1, the bottom strand of I_1 has its first toehold covered and the second toehold revealed. Signal Y then binds to I_1 at the second toehold, branch migrates to the $3'$ end of ^{+}Y and kicks

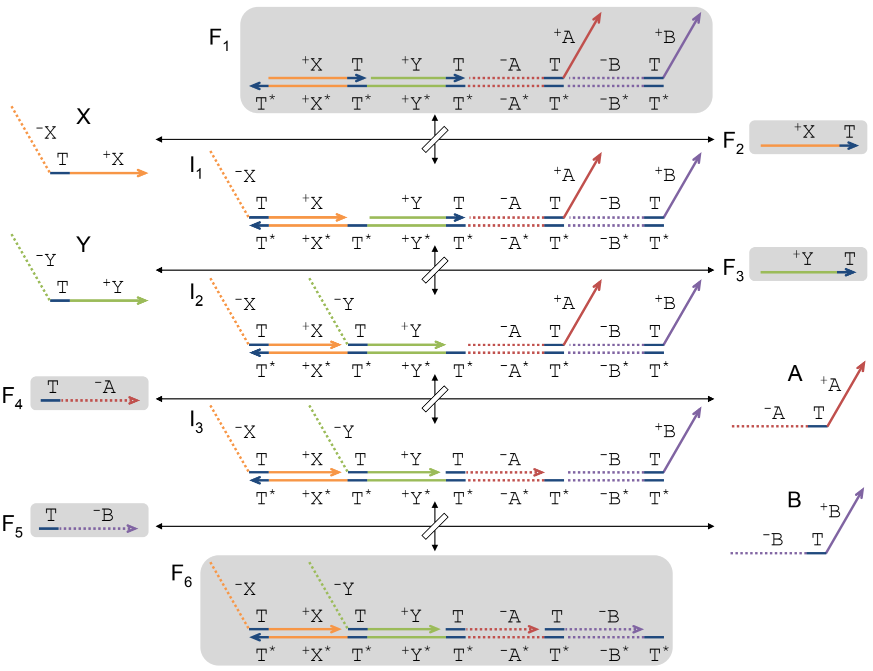


Fig. 2. The implementation of the formal bimolecular reaction $X + Y \rightleftharpoons A + B$ using history-free signal species X, Y, A, B. F_1 through F_6 are fuel species mediating the formal reaction, and are present in high concentration. F_1 and F_6 are unique to this reaction, while F_2 through F_5 may be shared with other reactions. I_1 through I_3 are intermediates of the formal reaction and they are all unique to this reaction. Reaction and domain notation is as in Fig. 1.

off the top strand F_3, producing intermediate I_2. The bottom strand of I_2 has its third toehold revealed and all the other toeholds covered. Now F_4 binds to I_2 at the third toehold, releases signal A and leaves intermediate I_3. F_5 binds to I_3 at the fourth toehold, releases signal B and leaves intermediate I_4. All the above reactions can be reversed by F_2, F_3, A and B reacting with I_1, I_2, I_3 and I_4 respectively. Finally, F_6 binds to I_4 at the last toehold and displaces the last top strand — which has no toehold domain by which to initiate the reverse reaction. Because of this last irreversible step, the overall reaction is irreversible.

If there is only signal Y but not X, nothing will happen because Y cannot directly react with any of the fuels. If there is only signal X but not Y, only the first step can happen and the backward reaction $F_2 + I_1 \rightarrow X + F_1$ ensures X is not permanently consumed. So, A and B will be produced, and X and Y consumed, only if both X and Y were initially present.

In Fig. 2, we simply remove domain I from F_1 in Fig. 1 to make the formal reaction reversible. Therefore, the F_6 of Fig. 1 becomes unnecessary and the final product of the forward reaction becomes the new F_6, which also serves as the first fuel of the backward reaction. Now there are only three intermediates instead of four. Note also that increasing the concentration of F_1 speeds up the forward reaction, while increasing the concentration of F_6 speeds up the backwards reaction — so this reaction can be individually tuned to be unbiased or biased forward or backward to any desired extent.

In considering the energy use of the reversible stack machine implementation it is important to verify that we are not cheating when we ignore the entropic contribution of the concentration changes of the fuels in the process of computation. Luckily, in the limit of large molecular counts of the fuel species, the contribution of the changing partial pressures of the fuels to the energy consumed in a reaction occurrence is independent of the history of the previous reaction events, and can be considered fixed for each reaction. Therefore we can use the fuels to provide a fixed forward bias for each reaction, or no bias if we start with equilibrium concentrations of the fuels.

Proper functioning of both the irreversible and reversible reaction modules involves a Brownian exploration of the system's state space to test if the preconditions for the reaction are met. Necessarily, this exploration must be reversible, in case the preconditions aren't met. This basic structure — reversible exploration to determine whether a subsequent step is possible — is used again and again in our constructions below. Despite the lack of determinism for individual steps, no species is incorrectly consumed or incorrectly produced — and so long as the system is biased forward (or unbiased), any species that should be produced will be produced eventually. In this sense, there can be no errors.

Creating new reactions between a set of signal species only requires the construction of appropriate fuel species and is thus programmable by the choice of fuel species without any alteration of the signal species themselves. Therefore, to implement a system of reactions, the fuels for each reaction may be individually constructed, and the union of these fuels correctly implements the full system.

We will consider stochastic dynamics for a small integral number of each signal species (typically a single copy each) reacting in a volume V in which the concentrations of fuel species are maintained constant. In a perfect implementation of a bimolecular reaction, the reaction rate should slow down exactly in proportion to V. It is easy to see that our multistep DNA implementation should have the same asymptotic scaling for kinetics. Consider the irreversible case. Among all the forward strand displacement reactions, only the second step (Y displacing F_3) is a bimolecular reaction between two low concentration species (the signal species and their reaction intermediates). The forward rate of this reaction (the second step) will scale as $1/V$. All the other steps are bimolecular reactions between one high concentration (fuel) species and one low concentration species and thus will be fast. Further, the backward reactions are either slow or (in the case of $F_3 + I_2 \rightarrow Y + I_1$) exactly balanced by a fast reaction ($F_4 + I_2 \rightarrow A + I_3$) in the forward direction. Thus, the second step is the rate limiting step for the reaction pathway. (A similar argument holds for both forward and backward reactions in the reversible scheme.)

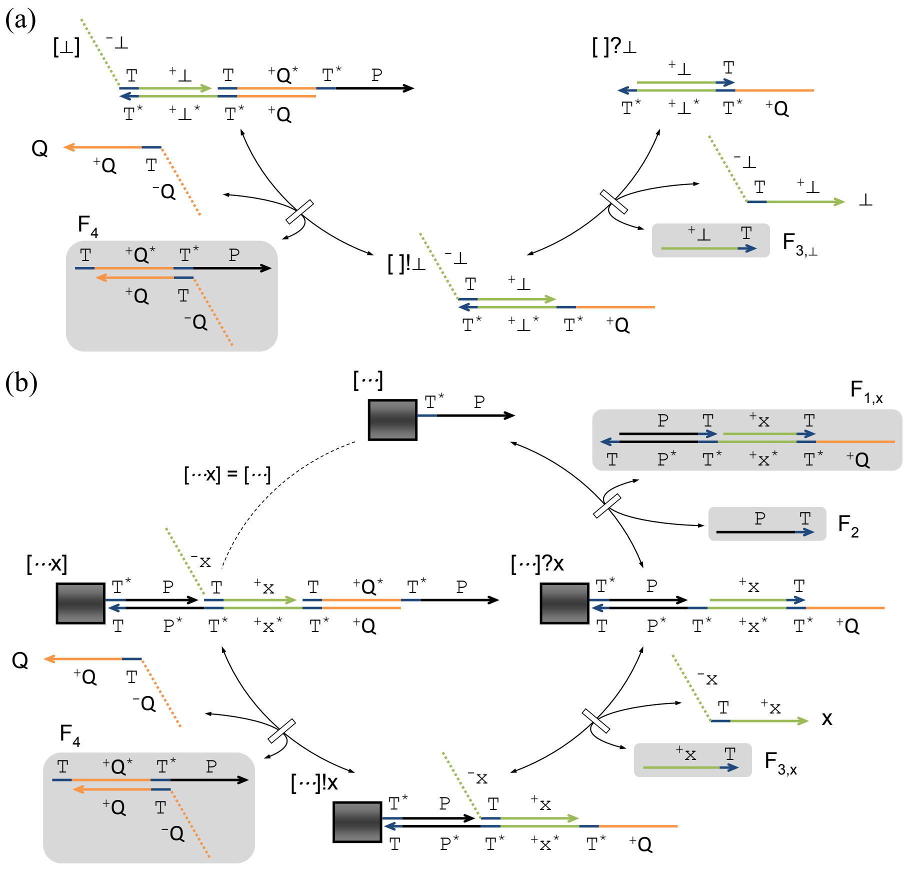


Fig. 3. The implementation of the formal polymer reaction $[\cdots] + x \rightleftharpoons [\cdots x] + Q$. Intuitively, in the forward direction this reaction adds a new monomer to the end of the polymer, releasing Q to signal completion. In the backward direction, Q detaches the last monomer from the polymer. **(a)** The beginning of the polymer and the strand displacement reactions when the stack is empty, i.e., implementation of reaction (1b). Note that the first monomer is $\perp$ to indicate the end of the stack for the stack machine simulation. **(b)** The strand displacement reactions for the monomer addition / removal cycle, i.e., implementation of reaction (1a). Clockwise: adding a new monomer; counter-clockwise: removing the last monomer. The dark box indicates the 'left' side of the polymer, with an arbitrary number of subunits. The dotted line indicates conceptually encapsulating the x subunit repeat block within the dark box. Reaction and domain notation is as in Fig. 1.

Unimolecular reactions such as $X \to A$ can be implemented analogously with the appropriate shortening or extension of fuel F_1, as can higher order reactions such as $X + Y + Z \to A + B + C$ and asymmetric reactions such as $X \to A + B$. (Similar modifications can be used for reactions $X \rightleftharpoons A$, $X + Y + Z \rightleftharpoons A + B + C$, $X \rightleftharpoons A + B$, etc.) Extending our construction to reactions of order n yields $n - 1$ forward steps that involve bimolecular reactions between two low concentration species. Therefore we would encounter a slowdown scaling as $1/V^{n-1}$, with the unimolecular implementation being independent of volume, as is required to agree with standard chemical kinetics.

3 A Reversible Polymer Addition Primitive

Finite CRNs by definition involve a finite number of possible molecular species, and this limits their behavioral complexity. Extending CRNs to include polymers allows one to give a finite specification of a molecular system involving potentially an infinite number of distinguishable species — polymers of different lengths and with different sequences — that interact according to a finite number of local rules. A variety of extensions of CRNs to polymers (and other combinatorial structures) have been considered, differing in the types of local rules (e.g. end-localized reactions, interior-localized reactions, polymer joining and scission) and the types of polymer structures (e.g. strictly linear, branched, networks with cycles) that are allowed [13,8,11]. All these natural CRN extensions can efficiently simulate Turing machines (c.f. [11]). However, whereas these extensions were designed to be general for modeling biochemical systems, our interest here is in a language for specifying polymer systems that can be implemented with DNA strand displacement reactions — and fully general modeling languages presently may be too difficult to compile into DNA.

Therefore, we focus on a very limited subset of polymer reactions that simultaneously allows implementation with DNA and is capable of efficient simulation of Turing machines. Specifically we make use of a single reversible reaction mechanism that (in the forward direction) appends a desired subunit onto the polymer while releasing a confirmation signal, and that (in the reverse direction) upon receipt of a query detaches a subunit from the polymer. In our construction, each polymer has a fixed end and a growing end. Appending and detaching can only occur on the growing end. All polymers begin at their fixed ends with a special subunit, $\perp$, followed by an arbitrary sequence of subunits from the finite set Σ. Formally, a polymer with sequence $\perp w$, where $w \in \Sigma^*$, is written as $[\perp w]$. The subunits themselves may also exist as free species, $x \in \Sigma$, as may the special subunit $\perp$. The query/confirmation species is called Q. Then the implemented reversible polymer addition reaction may be written as

$$[\cdots] + x \rightleftharpoons [\cdots x] + Q \tag{1}$$

where informally $[\cdots]$ represents a polymer with some sequence $\perp w$ and $[\cdots x]$ represents that same polymer extended to sequence $\perp wx$. Formally, this single polymer reaction schema represents an infinite family of specific reactions

$$[\perp w] + x \rightleftharpoons [\perp wx] + Q \tag{1a}$$

for all $w \in \Sigma^*$ and $x \in \Sigma$, as well as the base case

$$[\,]?\perp + \perp \rightleftharpoons [\perp] + Q \tag{1b}$$

that enables detecting that the polymer is a monomer.

Fig. 3 shows the DNA implementation of this polymerization primitive. The formal species Q and $\perp$ and each $x \in \Sigma$ are implemented as finite CRN species using the same history-free motif described in section 2. The polymer itself is a chain of information-bearing subunits (green) spliced together by strands using the P and ^{+}Q domains. To mediate the desired reactions, a number of high-concentration fuels are used: F_2 and F_4 are independent of the monomer type, while a separate $F_{1,x}$ and $F_{3,x}$ is needed for each $x \in \Sigma$, as well as $F_{3,\perp}$.

The reversible and exploratory nature of the DNA implementation's reactions are essential to its function. For example, for $[\cdots]$ to react with a specific x that may be present, it may first react with several different $F_{1,y}$ to produce the intermediate $[\cdots]?y$ that attempts to add y to the polymer — but so long as y is not present in solution, this attempt fails, and the reaction reverses to recreate $[\cdots]$ with the help of F_2. Eventually, $[\cdots]$ will react with $F_{1,x}$ to produce $[\cdots]?x$, which can react with x to proceed to the next step, the $[\cdots]!x$ intermediate. Finally a reaction with F_4 brings the polymer back to its canonical state, $[\cdots x]$, but with the new subunit appended and the confirmation Q produced. Of course, since all the reactions are reversible, Q can also serve as a query and reverse $[\cdots x]$ to $[\cdots]!x$, after which the appropriate $F_{3,x}$ may succeed in detaching x, so that F_2 can bring $[\cdots]?x$ back to $[\cdots]$. A set of different $F_{3,y}$ make sure the detaching will happen no matter what subunit is on the growing end of the polymer. For the base case, the reverse can only go as far as $[\cdots]?\perp$, which has a special form and thus won't be able to react with F_2. This ensures that $\perp$ is always the first subunit of the polymer.

While the polymer reactions by themselves do nothing more than push and pop subunits back and forth onto and off of the end of the polymer, these reactions can be controlled and driven by a simultaneously active finite CRN that interacts with the formal species Q, $\perp$, and $x \in \Sigma$. To append a subunit to the polymer, the CRN must simply produce the desired species x and then wait for the confirmation Q. To read a subunit off the polymer, the CRN must simply produce the query Q and then wait for the arrival of some subunit $x \in \Sigma$ or else $\perp$.

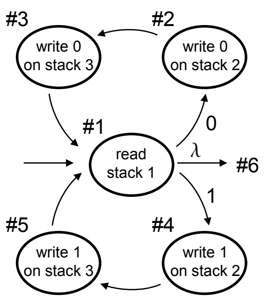
Our polymer reaction primitive is also essentially bimolecular, and hence the kinetics also scales as $1/V$.

4 Irreversible Stack Machine Implementation

4.1 Definition of Stack Machines

In this paper we provide a direct molecular implementation of stack machines rather than the more familiar Turing machines because they are particularly matched to the kind of polymer operation we have available, which accesses the polymer at one end only. The stack machine model of computation intuitively consists of a finite state control together with memory in the form of a finite number of stacks. Each stack can hold an arbitrary sequence of symbols but can only be accessed at one end: stack operations include pushing a new symbol onto a stack, or popping a symbol off a stack, as well as detecting an empty stack.

(a)



(b)

$\#1\ 0\ 1 \longrightarrow \#2$
$\#1\ 1\ 1 \longrightarrow \#4$
$\#1\ \lambda\ 1 \longrightarrow \#6\ \lambda\ 1$
$\#2 \longrightarrow \#3\ 0\ 2$
$\#3 \longrightarrow \#1\ 0\ 3$
$\#4 \longrightarrow \#5\ 1\ 2$
$\#5 \longrightarrow \#1\ 1\ 3$

(c)

1. (#1, 00111, λ, λ)
2. (#4, 0011, λ, λ)
3. (#5, 0011, 1, λ)
4. (#1, 0011, 1, 1)
5. (#4, 001, 1, 1)
6. (#5, 001, 11, 1)
7. (#1, 001, 11, 11)
8. (#4, 00, 11, 11)
9. (#5, 00, 111, 11)
10. (#1, 00, 111, 111)
11. (#2, 0, 111, 111)
12. (#3, 0, 1110, 111)
13. (#1, 0, 1110, 1110)
14. (#2, λ, 1110, 1110)
15. (#3, λ, 11100, 1110)
16. (#1, λ, 11100, 11100)
17. (#6, λ, 11100,11100)

(d)

$S_{\#1} + 0_1 \longrightarrow S_{\#2} + Q$
$S_{\#1} + 1_1 \longrightarrow S_{\#4} + Q$
$S_{\#1} + \perp_1 \longrightarrow S_{\#6} + \perp_1$
$S_{\#2} + Q \longrightarrow S_{\#3} + 0_2$
$S_{\#3} + Q \longrightarrow S_{\#1} + 0_3$
$S_{\#4} + Q \longrightarrow S_{\#5} + 1_2$
$S_{\#5} + Q \longrightarrow S_{\#1} + 1_3$

$Q_1 \longleftrightarrow Q$
$Q_2 \longleftrightarrow Q$
$Q_3 \longleftrightarrow Q$

$[\ldots]_1 + 0_1 \longleftrightarrow [\ldots 0]_1 + Q_1$
$[\ldots]_1 + 1_1 \longleftrightarrow [\ldots 1]_1 + Q_1$
$[\ldots]_2 + 0_2 \longleftrightarrow [\ldots 0]_2 + Q_2$
$[\ldots]_2 + 1_2 \longleftrightarrow [\ldots 1]_2 + Q_2$
$[\ldots]_3 + 0_3 \longleftrightarrow [\ldots 0]_3 + Q_3$
$[\ldots]_3 + 1_3 \longleftrightarrow [\ldots 1]_3 + Q_3$

(e)

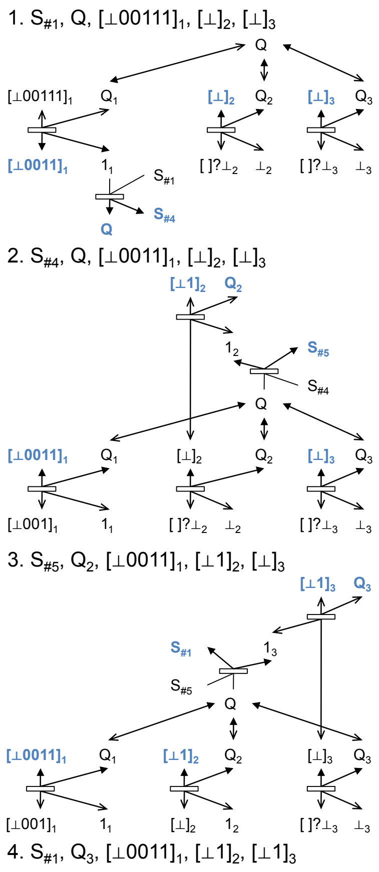


Fig. 4. Example execution of a stack machine program. **(a)** Diagrammatic representation of a stack machine that reads a string on stack 1 and writes a reversed copy onto both stack 2 and stack 3. **(b)** Transition rules for the same stack machine. **(c)** Execution history of stack machine configurations for computation with input string 00111. **(d)** Polymer CRN reactions for the same stack machine. Recall that polymer reaction schema of form (1) expand to reactions of forms (1a) and (1b). **(e)** Reaction pathways within the polymer CRN implementation, illustrated for the first three steps going from configuration 1 to configuration 4. Solid arrowheads indicate the direction of computation that is ratcheted forward by this step's irreversible reaction, and blue species represent the canonical endpoint species for each step of the computation. Each reaction is shown with longer polymer species on top, which is why in some steps reactions go "up" and in other steps reactions go "down".

Input is provided as the initial sequence of symbols in the first stack. While stack machines with only 1 stack are known to be less than Turing universal, 2 stacks are enough for universality. Similarly, while stack machines with just 1 symbol (also known as counter machines or register machines) are universal [17], they are exponentially slower than Turing machines, and efficient simulation of

Turing machines becomes possible only with 2 symbols or more. In fact, multi-stack multi-symbol stack machines can simulate multi-tape multi-symbol Turing machines with no slow-down (and vice versa). Consequently, many stacks and many symbols are preferred for elegance and efficiency. This is what we achieve with our DNA polymer implementation.

We allow any finite alphabet of symbols Σ, with an additional symbol $\lambda \notin \Sigma$ to indicate that the stack is empty. We specify stack machine transition rules in a somewhat non-standard manner — one that is better suited to discussing reversibility (see next section). There are 4 types of transition rules:

$$\begin{array}{ll}
1. & \alpha\, x\, i \longrightarrow \beta\, y\, j \\
2. & \alpha\, x\, i \longrightarrow \beta \\
3. & \alpha \qquad \longrightarrow \beta\, y\, j \\
4. & \alpha\, \lambda\, i \longrightarrow \beta\, \lambda\, i
\end{array}$$

where α, β are states, $x, y \in \Sigma$ are symbols, and $i, j \in \{1, \ldots, n\}$ are stacks. A transition rule of type (1) means: when in state α and the top symbol on stack i is x, pop it off and push symbol y onto stack j, transitioning to state β. A transition rule of type (2) means: when in state α and the top symbol on stack i is x, pop it off and transition to state β. A transition rule of type (3) means: when in state α, push symbol y onto stack j, transitioning to state β. A transition rule of type (4) means: when in state α and stack i is empty, move to state β. Note that the stack on the left and right must be the same for rule type (4).

A configuration of a stack machine consists of a state α and the contents of stacks $1, \ldots, n$. Computation begins in the designated start state (typically #1) with the input to the computation on the stacks (typically stack 1 has an input string, and the remaining stacks are empty) and proceeds by execution of applicable rules until no rule is applicable (typically, the machine will be in a 'halting state' that does not appear in the LHS of any rule). The contents of the stacks after halting may be considered the output of the machine.

We say that the stack machine is (syntactically) deterministic if for every configuration, there is at most one applicable transition rule. This can easily be verified by checking that for each state α, either all rules with α on the LHS read from the same stack with at most one transition per read symbol, or else there is at most one rule of type (3).

An example stack machine and computation is shown in Fig. 4abc.

4.2 Reactions Corresponding to the Transition Rules

The polymer CRN implementation of an n-stack machine with symbol alphabet Σ will comprise a finite collection of CRN reactions, one for each transition rule and one for each stack, combined with a polymer reaction for each stack. We require n distinct types of polymer reactions to implement the n stacks. We obtain them by generating n independent copies of the polymer reaction primitive of Fig. 3 wherein every species and domain is subscripted by i to indicate that the domains are unique to that polymer type (with the exception of the universal toehold T). Thus the fuels are also unique to the polymer type. Because of the unique domains and fuels, the reactions steps of Fig. 3 will never result in crosstalk between polymers of different type. Therefore, generating x_i or Q_i will result in pushing x onto or popping a symbol off of stack i specifically. Later it will be convenient to have a single 'query' species Q that interconverts with the Q_i to (reversibly) read any stack; the symbol x_i that is read indicates which stack it came from, so no information is lost.

In summary, for every stack $i \in \{1, \ldots, n\}$ and symbol $x \in \Sigma$, we have a distinct molecular species x_i. Further, for every stack i we have species $\perp_i$, the CRN reaction

$$Q \rightleftharpoons Q_i$$

and the polymer reaction

$$[\cdots]_i + x_i \rightleftharpoons [\cdots x]_i + Q_i.$$

If a polymer contains only the symbol $\perp_i$ then the polymer represents stack i being empty, i.e. having content λ.

To run the system, we start with (1) exactly one molecule of each stack polymer type, each containing the input strings for its corresponding stack; (2) exactly one molecule of the state species $S_{\#1}$; and (3) exactly one molecule of the 'query' species Q, which rapidly interconverts into the Q_i required for each stack. Our system will respect the conserved property that there is always either exactly one Q or one Q_i molecule for some stack, or else there is exactly one molecule x_i representing some symbol on some stack.

Now, each stack machine transition rule 1-4 corresponds to a single polymer CRN reaction as follows:

$$\begin{array}{llcl}
1. & \alpha\, x\, i \longrightarrow \beta\, y\, j & \Rightarrow & S_\alpha + x_i \rightarrow S_\beta + y_j \\
2. & \alpha\, x\, i \longrightarrow \beta & \Rightarrow & S_\alpha + x_i \rightarrow S_\beta + Q \\
3. & \alpha \qquad \longrightarrow \beta\, y\, j & \Rightarrow & S_\alpha + Q \rightarrow S_\beta + y_j \\
4. & \alpha\, \lambda\, i \longrightarrow \beta\, \lambda\, i & \Rightarrow & S_\alpha + \perp_i \rightarrow S_\beta + \perp_i
\end{array}$$

Illustrative steps for the implementation of the example stack machine are shown in Fig. 4de. Note that despite all the reversible reactions, each time an irreversible CRN reaction (corresponding to a transition rule) occurs, the overall computation ratchets forward.

Run in a reaction volume V, each irreversible step will take average time $O(V)$ since its rate scales as $O(1/V)$. So to simulate a stack machine (or TM) whose computation runs in time t using space s, our DNA implementation will take time $O(tV)$. However, the reaction volume must be large enough to contain the polymers, which will be $O(s)$ subunits long, and hence $V = O(s)$. Taking the worst-case bound $s = O(t)$, the overall time required by the DNA stack machine implementation is $O(t^2)$. In contrast, because Bennett's hypothetical polymer-chemistry Turing machine has no bimolecular reactions between low-concentrations species (all reactions are between a single polymer tape and high-concentration enzymes), its time requirement is just $O(t)$ — better than ours.

5 Reversible Stack Machine Implementation

Given a stack machine defined as in the preceding section, the set of reverse rules is formed by switching the left-hand side and the right-hand side of all rules. We say the stack machine is reversible if the set of reverse rules is deterministic.

To implement a stack machine that can proceed either forward or backward in chemistry we can use reversible reactions:

$$\begin{array}{lllll}
1. & \alpha\, x\, i \longrightarrow \beta\, y\, j & \Rightarrow & S_\alpha + x_i \rightleftharpoons S_\beta + y_j \\
2. & \alpha\, x\, i \longrightarrow \beta & \Rightarrow & S_\alpha + x_i \rightleftharpoons S_\beta + Q \\
3. & \alpha \longrightarrow \beta\, y\, j & \Rightarrow & S_\alpha + Q \rightleftharpoons S_\beta + y_j \\
4. & \alpha\, \lambda\, i \longrightarrow \beta\, \lambda\, i & \Rightarrow & S_\alpha + \bot_i \rightleftharpoons S_\beta + \bot_i
\end{array}$$

5.1 Simulating a Reversible Turing Machine

Most theoretical work on reversible computing uses Turing machines rather than stack machines. Are there non-trivial reversible stack machines according to the above definition? Are there universal reversible stack machines according to the above definition?

We can take the path of showing that reversible stack machines can simulate known reversible Turing machines. For simplicity let us consider a binary, reversible Turing machine with one tape that is bounded on the left and is infinite on the right; futher, we require that the Turing machine never tries to read past the left end of the tape. (For example ref. [18] describes such a Turing machine that is universal, although it is slow; multi-tape reversible Turing machines are faster, and can be similarly implemented with polymer CRNs.) We can consider three types of Turing machine transition rules, using Bennett's notation [5]:

$$\begin{array}{ll}
1. & \alpha\, x \longrightarrow \beta\, y \\
2. & \alpha\, / \longrightarrow \beta - \\
3. & \alpha\, / \longrightarrow \beta +
\end{array}$$

where α, β are states, and $x, y \in \{0, 1\}$ are symbols. The first rule means that when in state α with the head reading symbol x, transition to state β overwriting x with y. The second and third rules indicate that when in state α move left or right respectively, without reading from or writing to the tape. The reverse of rule $\alpha\, x \longrightarrow \beta\, y$ is $\beta\, y \longrightarrow \alpha\, x$. The reverse of rule $\alpha\, / \longrightarrow \beta-$ is $\beta\, / \longrightarrow \alpha+$ and vice versa.

We represent the tape using two stacks. Everything to the left of the head is on stack 1 with the current symbol on top. Everything to the right of the head is on stack 2 with the symbol to the right of the head on top. The infinity of 0's past the rightmost 1 on the Turing machine tape is implicitly represented such that the topmost symbol on stack 2 can only be 1, if the stack is not empty. We convert Turing machine transition rules to stack machine transition rules as follows, where $x, y \in \{0, 1\}$:

1. $\alpha\, x \longrightarrow \beta\, y \quad \Rightarrow$

$$\alpha\, x\, 1 \longrightarrow \beta\, y\, 1$$

2. $\alpha\, / \longrightarrow \beta- \quad \Rightarrow$

$$\begin{array}{lcl}
\alpha \;\; 0\,1 & \longrightarrow & \sigma_1\; 0\,1 \\
\alpha \;\; 1\,1 & \longrightarrow & \beta \;\; 1\,2 \\
\sigma_1\; \lambda\,2 & \longrightarrow & \sigma_2\; \lambda\,2 \\
\sigma_1\; 0\,2 & \longrightarrow & \sigma_4\; 0\,2 \\
\sigma_1\; 1\,2 & \longrightarrow & \sigma_4\; 1\,2 \\
\sigma_2\; 0\,1 & \longrightarrow & \sigma_3 \\
\sigma_3\; \lambda\,2 & \longrightarrow & \beta \;\; \lambda\,2 \\
\sigma_4\; 0\,1 & \longrightarrow & \beta \;\; 0\,2
\end{array}$$

3. $\alpha\, / \longrightarrow \beta+ \quad \Rightarrow$

$$\begin{array}{lcl}
\alpha \;\; 0\,2 & \longrightarrow & \sigma_4\; 0\,1 \\
\alpha \;\; 1\,2 & \longrightarrow & \beta \;\; 1\,1 \\
\alpha \;\; \lambda\,2 & \longrightarrow & \sigma_3\; \lambda\,2 \\
\sigma_4\; 0\,2 & \longrightarrow & \sigma_1\; 0\,2 \\
\sigma_4\; 1\,2 & \longrightarrow & \sigma_1\; 1\,2 \\
\sigma_3 & \longrightarrow & \sigma_2\; 0\,1 \\
\sigma_2\; \lambda\,2 & \longrightarrow & \sigma_1\; \lambda\,2 \\
\sigma_1\; 0\,1 & \longrightarrow & \beta \;\; 0\,1
\end{array}$$

where σ_1 — σ_4 are states unique to the given Turing machine transition rule. Note that (3) is the reverse of (2). The hard work involved in moving left and right comes from the requirement to maintain a consistent and unique implicit representation of the infinite background of zeros on the right.

It is enough to prove two things: that the forward direction is deterministic, and that the forward direction correctly simulates the Turing machine transitions. Then no point can have multiple predecessors because simulating the reverse Turing machine transition crosses that point in the opposite direction. (The fact that the forward and backward stack machine paths within a single Turing machine transition must be the same follows from the fact that for the reverse path we could have just applied the forward rules in reverse order.)

Forward determinism follows because in any stack machine state we are reading at most one stack. It is also easy to verify that (2) and (3) correctly simulate the Turing machine transition in the forward direction by following all the branches. For that, note that (2) guarantees that the bottom symbol in stack 2, if any, is a 1. (I.e.: no unnecessary "blanks". We also assume this is true of the initial state.) Thus in (3), stack 2 cannot be empty when we get to state σ_4.

More efficient reversible Turing machines with multiple tapes and large alphabets can be simulated in a similar manner, as a straightforward generalization of the given construction. This is important because whereas 1-tape, 2-symbol reversible Turing machines are indeed universal, multi-tape Turing machines are essential for Bennett's theorems showing that the time and space requirements for logically reversible Turing machine computation are no more than slightly worse than linear with respect to irreversible Turing machine computation [5,7].

6 Conclusions

Our paper contributes to the art of designing molecular interactions using strand displacement cascades by proposing a direct implementation of arbitrary coupled reversible reactions, as well as a way to add and remove end monomers to and from a DNA polymer. The new construction for reversible reactions is more efficient than implementing them as two separate irreversible reactions [25], both in terms of the complexity of the scheme as well as the amounts of fuel reagents required. By adjusting and maintaining fuel concentrations, reactions can be biased forward or backward or balanced arbitrarily close to equilibrium. Based on these reaction mechanisms, we developed a novel method of embedding computation in biochemical and biological systems by showing an efficient autonomous stack machine simulation. This simulation can be made reversible to attain low energy consumption.

Different architectures for molecular computing such as algorithmic self-assembly, circuits implemented with CRNs, Turing machines implemented with CRNs, and polymer CRNs embody different tradeoffs between time, volume, energy and uniformity. Our construction is exponentially more efficient in terms of the required molecular counts and volume than geometry-free Turing-universal computation using strand displacement reactions (combination of refs. [25] and [24]), and also polynomially faster. Moreover, unlike the geometry-free computation of ref. [24], our polymer CRN construction in theory yields the correct computation output with probability 1.

Lastly, using our implementation of reversible CRNs, we proposed a logically-reversible stack machine construction that maintains error-free computation using physically reversible reactions. We showed that these reversible stack machines can reversibly simulate a reversible Turing machine, establishing their Turing universality and the applicability of results in the existing literature. Our constructions can be viewed as steps toward a DNA implementation of Bennett's thought experiment [6] in which computation was shown to require arbitrarily little thermodynamic energy per step.

There is still room for improvement in our constructions. First, the fact that the machine state is stored within multiple free-floating molecules results in the requirement for slow bimolecular reactions, unlike Bennett's hypothetical scheme. A second drawback of our scheme is that preparing reactions with a single copy of each state-bearing molecule would be difficult experimentally, but our system will not correctly simulate stack machines if run with multiple copies of state or stack molecules, or if run with mass action. Further, unlike Bennett's scheme, ours cannot run an arbitrary number of parallel machines in the same reaction chamber. This limitation prevents the use of our construction for fast solutions to parallel search problems [1].

Finally, our construction lacks the attractive feature of material recycling: taking any irreversible Turing machine, applying the transform described in ref. [5] to make it reversible, and implementing it with Bennett's hypothesized molecular construction, yields a molecular computation that recycles all material requirements except for the molecules used in writing out the output. However, in our scheme, different fuel molecules would be used in the "compute" and "retrace" phases of the transformed Turing machine computation, and would not be regenerated. Indeed, every computation converts fuels of forward reactions to the fuels of reverse reactions in an amount proportional to the length of the computation.

The polymer reactions we introduce are likely instances of a wider class of polymer modification reactions that can be implemented with strand displacement. It would be exciting to implement a polymer reaction class capable of exhibiting the richness of cytoskeletal networks that are responsible for cellular reorganization and coordinated movement.

Acknowledgments

We thank Ho-Lin Chen for insightful discussions and suggestions. Our development of the history-free CRN scheme grew out of extensive discussions with Luca Cardelli. We thank Anne Condon for clarifying discussions. This work was supported by the Molecular Programming Project under NSF grant 0832824 and an NSF CIFellows Award to DS.

References

1. Adleman, L.: Molecular Computation of Solutions to Combinatorial Problems. Science 266(5187), 1021–1024 (1994)
2. Beaver, D.: A Universal Molecular Computer. In: Lipton, R., Baum, E. (eds.) DNA Based Computers, pp. 29–36. AMS, Providence (1996)
3. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules. Nature 414(6862), 430–434 (2001)
4. Benenson, Y., Shapiro, E.: Molecular computing machines. In: Encyclopedia of Nanoscience and Nanotechnology, pp. 2043–2056 (2004)
5. Bennett, C.: Logical reversibility of computation. IBM Journal of Research and Development 17(6), 525–532 (1973)
6. Bennett, C.: The thermodynamics of computation – a review. International Journal of Theoretical Physics 21(12), 905–940 (1982)
7. Bennett, C.: Time/space trade-offs for reversible computation. SIAM Journal on Computing 18, 766–776 (1989)
8. Blinov, M., Faeder, J., Goldstein, B., Hlavacek, W.: BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Bioinformatics 20(17), 3289–3291 (2004)
9. Cardelli, L.: Strand algebras for DNA computing. In: Deaton, R., Suyama, A. (eds.) DNA 15. LNCS, vol. 5877, pp. 12–24. Springer, Heidelberg (2009)
10. Cardelli, L.: Two-Domain DNA Strand Displacement. In: Developments in Computational Models (DCM), pp. 33–47 (2010)
11. Cardelli, L., Zavattaro, G.: On the computational power of biochemistry. In: Horimoto, K., Regensburger, G., Rosenkranz, M., Yoshida, H. (eds.) AB 2008. LNCS, vol. 5147, pp. 65–80. Springer, Heidelberg (2008)
12. Chen, H., De, A., Goel, A.: Towards Programmable Molecular Machines. In: Foundations of Nanoscience (FNANO) (2008)

13. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-based modelling of cellular signalling. In: Caires, L., Li, L. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 17–41. Springer, Heidelberg (2007)
14. Kurtz, S., Mahaney, S., Royer, J., Simon, J.: Biological computing. In: Complexity Theory Retrospective II, pp. 179–195 (1997)
15. Landauer, R.: Irreversibility and heat generation in the computing process. IBM Journal of Research and Development 5(3), 183–191 (1961)
16. Liekens, A.M.L., Fernando, C.T.: Turing complete catalytic particle computers. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 1202–1211. Springer, Heidelberg (2007)
17. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Englewood Cliffs (1967)
18. Morita, K., Shirasaki, A., Gono, Y.: A 1-tape 2-symbol reversible Turing machine. The Transactions of the IEICE E 72(3), 223–228 (1989)
19. Qian, L., Winfree, E.: A simple DNA gate motif for synthesizing large-scale circuits. In: Goel, A., Simmel, F.C., Sosík, P. (eds.) DNA 14. LNCS, vol. 5347, pp. 70–89. Springer, Heidelberg (2009)
20. Rothemund, P.: A DNA and restriction enzyme implementation of Turing machines. In: Lipton, R., Baum, E. (eds.) DNA Based Computers. DIMACS, vol. 27, pp. 75–119. AMS, Providence (1996)
21. Rothemund, P., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA sierpinski triangles. PLoS Biology 2(12), e424 (2004)
22. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)
23. Smith, W.: DNA computers in vitro and vivo. In: Lipton, R., Baum, E. (eds.) DNA Based Computers. DIMACS, vol. 27, pp. 121–185. AMS, Providence (1996)
24. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Natural Computing 7(4), 615–633 (2008)
25. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. Proceedings of the National Academy of Science 107(12), 5393–5398 (2010)
26. Winfree, E.: On the computational power of DNA annealing and ligation. In: Lipton, R., Baum, E. (eds.) DNA Based Computers. DIMACS, vol. 27, pp. 199–221. AMS, Providence (1996)
27. Winfree, E.: Simulations of computing by self-assembly. Technical Report CS-TR:1998.22, Caltech (1998)
28. Yin, P., Turberfield, A., Sahu, S., Reif, J.: Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 10. LNCS, vol. 3384, pp. 426–444. Springer, Heidelberg (2005)
29. Zhang, D., Turberfield, A., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318(5853), 1121–1125 (2007)
30. Zhang, D., Winfree, E.: Control of DNA strand displacement kinetics using toehold exchange. Journal of the American Chemical Society 131(47), 17303–17314 (2009)

Reversible Transition of Photonic DNA Automaton Using Hairpin-DNA Responding to a Single Kind of Photonic Signal

Hiroto Sakai, Yusuke Ogura, and Jun Tanida

Department of Information and Physical Sciences,
Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
{h-sakai,ogura,tanida}@ist.osaka-u.ac.jp

Abstract. In this paper, we report a scheme for reversible state-transition of the automaton responding to a single kind of a photonic signal. The state is encoded into the conformation of hairpin-DNAs tethered with azobenzene, which is responsive to visible and UV light irradiation. The reversible behavior is realized by carrying out a sequence of conformation-change of the hairpin-DNA in a stepwise manner. Experimental results demonstrate that the state can be changed reversibly according to photonic signals.

1 Introduction

DNA automaton is a computing basis for autonomous and sequential processing at a molecular scale[1]. DNA automaton is useful for in-situ measurement of states of particular molecules in a molecular system and for control of the molecules according to the environmental conditions. For example, E. Shapiro *et al*. demonstrated programmed release of a single-stranded DNA that works as a drug using DNA automaton[2].

Treatment of molecules without changing the environment where they exist is important for in-situ measurement or control because it enables to deal with the original information on the molecules. To control the automaton from out of the solution, external signaling is required. Adding molecules and changing temperature are good examples of the external signaling; however, these strategies can induce significant change of the environmental conditions. Light is a promising carrier for external signaling because it enables to carry the information in parallel and locally. Flexible manipulation of the light is achievable using photonic devices such as a spatial light modulator. Utilizing optical signals as an external one, we can control molecules remotely without changing the environmental conditions.

We are studying about photonic DNA automaton, which is a computational paradigm by using light and DNA as information carriers[3]. Photonic methodology enables a sequential control of molecules in small volumes in parallel and remotely, and offers control of biomolecular systems with external signals. In a previous paper, we demonstrated photonic switch of a bistable structure using a hairpin-DNA tethered with azobenzene[4]. The hairpin-DNA changes to the open-state with ultra-violet (UV) light irradiation and to the closed-state with visible light irradiation owing to photo-isomerization of the azobenzene. Moreover, we designed another hairpin-DNA operating the opposite manner: the hairpin-DNA forms the open-state (the closed-state) with visible (UV) light irradiation. The destination-state of these types of the hairpin-DNAs depends only on the input but independent on the current state.

This paper reports on a scheme for a reversible and stepwise state-transition responding to a single kind of an input signal. The single input signal consists of a sequence of light irradiations. We show the reversible state-transition experimentally. Section 2 explains a proposed scheme. In Section 3, we show experimental results. In Section 4, we give conclusions.

2 Reaction Schemes

The state-transition diagram considered in this study is shown in Fig. 1(a). The internal state changes to the other state for input "1" and stays for input "0". We designed a DNA reaction scheme using a hairpin-DNA. The structure of the hairpin-DNA represents the internal-state of the automaton, and the conformational change corresponds to its transition. The hairpin-DNA changes between the closed-state (S_0) and the open-state (S_1) owing to photo-isomerization of azobenzene[5] by UV and visible light irradiation. Input 1 is encoded into a sequential irradiation of UV and visible light, and input 0 is non-irradiation(Fig. 1(a)). The number of input symbols is limited by two because possible number of the form of the azobenzene is two.

Four kinds of DNA strands were used in the scheme. Strands O called opener, C called closer and U were tethered with azobenzene, and they form different structures under light irradiation as shown in Fig. 1(b). The structure of O consists of two loop-structures: left one is for control of reaction with hairpin-DNA and the right one is used for control of reaction with C. A strand U is used to open the right loop-structure. Hairpin-DNA opens by binding with O, and the open-state returns to a loop-structure by using C. The function of strands O and C are controlled according to a single kind of irradiation.

Figure 1(c) shows the designed scheme. Let us assume that the initial state is the closed-state of a hairpin-DNA, representing S_0. Under UV light irradiation, the azobenzene becomes cis-form and the left loop-structure of strand O opens. Hairpin-DNA can bind with the toe-hold, which is exposed as a result of the reaction. Then, azobenzenes are changed to the trans-form by visible light irradiation, and the loop of the hairpin-DNA opens. As a result, the internal-state changes from S_0 to S_1.

Switching form S_1 to S_0 (the lower part of Fig. 1(c)) is executed by a sequential binding of strands U and C in response to the same light irradiation (UV then visible) used for switching from S_0 to S_1(Fig. 1(c)). When the hairpin-DNA forms the structure shown at the upper right of Fig. 1(c), the function of the toe-hold

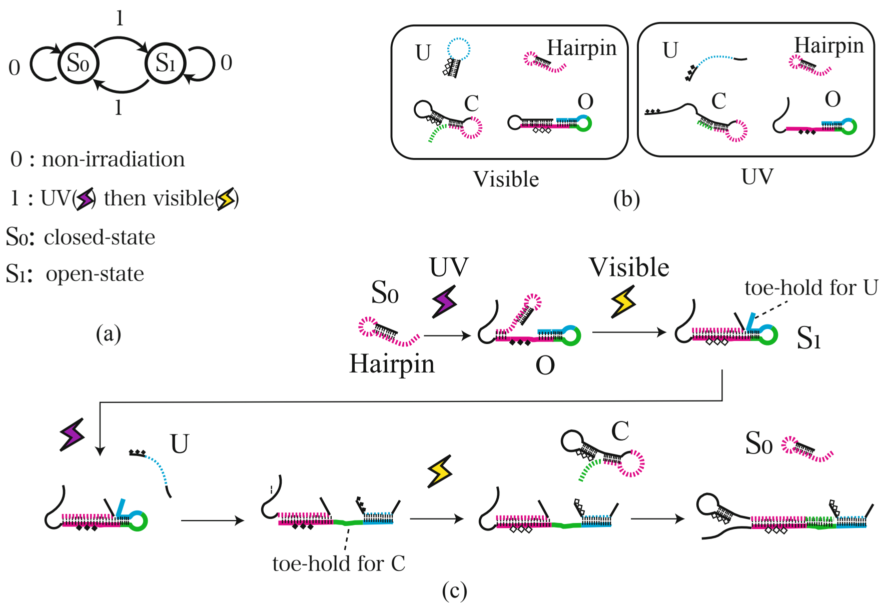


Fig. 1. (a)The state-transition diagram of considered automaton, (b)the structures under light irradiation and (c)the designed scheme

for binding with U is expressed owing to the displacement between the hairpin-DNA and a part of the right loop-structure of strand O. By irradiating UV light, the right loop-structure opens through the binding with U, then the toe-hold for binding with C is exposed. Under visible light irradiation, the hairpin-DNA is displaced by C as a result of binding between O and C, and the hairpin-DNA returns back to S_0.

3 Experiments

The reversible behavior was confirmed experimentally. The state after each operation was detected by measuring fluorescent intensity from a FRET pair labeled at the ends of hairpin-DNA. The DNA sequences used in experiments are shown in Fig. 2(a). A character "x" represents a single azobenzene. The sequences were designed using a software called NUPACK[6].

First, responses of strands C and U to light irradiation were investigated by measuring absorbance of the strands at 260 nm and 350 nm. Absorbance at 260 nm changes depending on the state of DNA, and the absorbance of a single-stranded DNA is higher than that of a double-stranded DNA. In contrast, absorbance at 350 nm depends on the form of the azobenzene. The absorbance of the trans-form azobenzene is larger than that of the cis-form azobenzene. The absorbance of both C and U increased at 260 nm and decreased at 350 nm after UV light irradiation. The result confirmed conformation-change of the strands with UV and visible light irradiations.

Hairpin

5'(FAM)-ACTCAACTTCACCGTGAAG-(BHQ)3'

O

5'-CCGCATCGCAGAGCGAAACGGTGAAGTTG
AGTGXATXACTTGCCGTAXTCXATATAXACXTT-3'

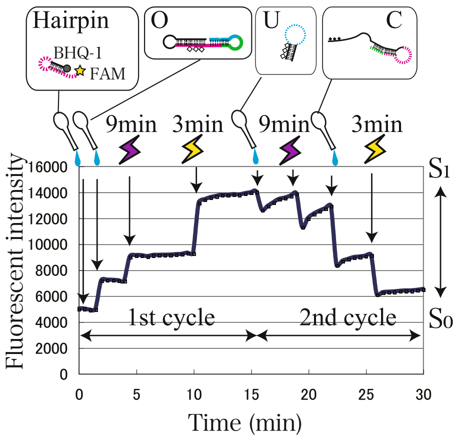
U

5'-GCGATGCGG-3'

C

5'-CGXCACXATTTTTXGGXCACCTAGCXCAXAACGT
AGCGTTCGGACCTCAACTTCACCGTTTCGC-3'

(a)



(b)

Fig. 2. (a) Sequences used in a experiment and (b) the experimental result

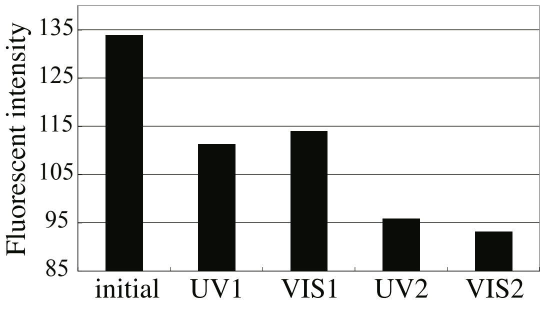


Fig. 3. Fluorescence intensities measured during a cycle of UV and visible light irradiation

Next, we executed reversible switch by adding DNA and irradiating light. Though the scheme is expected to work only by light irradiation, this operation strategy is used for easy operation. Figure 2(b) shows the fluorescence intensity measured during two cycles of UV and visible light irradiation. Irradiation time for UV light was 9 min, and the time for visible light was 3 min. These times are ones that photo-isomerization of azobenzene is saturated. The result suggests that a hairpin-DNA changes form S_0 to S_1 after the first cycle of irradiation, then returned back to S_0 after the second cycle of irradiation.

In the next experiment, we performed state-transition without additional DNA; namely we operated automaton only by optical signaling. In mixing solutions of four strands, strand O reacted with strand C unexpectedly. To inhibit the reaction, we redesigned strands O and C. The loop-structures in new strands

O and C become more stable, so that unexpected annealing between O and C can be suppressed. The sequence of redesigned O is CCGCATCGGAGACC-GATGCGGTGTxAAxTGxTTGAGGACxATxTAxCA, and that of redesigned C is ATxTTxGCTGATGCGGCTCAACATTACACCGCATCAG (The left and right edges are 5' and 3' -end, respectively). On the basis of the conformational stability of these strands, we changed irradiation time (10 min for visible light and 15 min for UV light) and the reaction temperature (45 degrees C).

Figure 3 shows the fluorescence intensity measured during two cycles of UV and visible light irradiation without adding DNA. Increase of the signal after first visible light irradiation and decrease after the second visible light irradiation suggests that a partial amount of the automaton ran intended transition. However, decrease of the signal after UV light irradiation shows that a number of hairpin-DNAs returned back to the closed-state with UV light irradiation. The hairpin-DNA changed to the open-state incompletely due to the azobenzene on strand O, which caused inhibition of the following reactions. To suppress the return back to the closed-state, we need to consider the number and the positions of azobenzenes tethered to the nucleotides.

4 Conclusions

A reversible state-transition of DNA automaton responding to a single kind of photonic signal was studied. We succeeded in executing state-transition reversibly by light irradiation with adding DNA. An experimental result also suggested that a partial amount of the automata were able to be driven by only photonic signaling (without adding DNA). Repeated operations of the automaton using photonic signals are expected to achieve by refinement of the scheme.

Acknowledgments

This work was supported by the Ministry of Education, Culture, Sports, Science and Technology, Japan, a Grant-in-Aid for Young Scientists (B) 20700276, 2008-2009, and by the Global COE Program "in silico medicine" at Osaka University.

References

1. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules. Nature 414, 430–434 (2001)
2. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expression. Nature 429, 1–6 (2004)
3. Sakai, H., Ogura, Y., Tanida, J.: Positional state representation and its transition control for photonic DNA automaton. In: Deaton, R., Suyama, A. (eds.) DNA 15. LNCS, vol. 5877, pp. 126–136. Springer, Heidelberg (2009)

4. Sakai, H., Ogura, Y., Tanida, J.: Implementation of a Nanoscale Automaton Using DNA Conformation Controlled by Optical Signals. Japanese Journal of Applied Physics 48, 09LA01 (2009)
5. Asanuma, H., Liang, X., Nishioka, H., Matsunaga, D., Liu, M., Komiyama, M.: Synthesis of azobenzene-tethered DNA for reversible photo-regulation of DNA functions: hybridization and transcription. Nature Protocols 2, 203–213 (2007)
6. http://www.nupack.org/

Simple Evolution of Complex Crystal Species

Rebecca Schulman[1] and Erik Winfree[2]

[1] University of California Berkeley, Berkeley, CA 94720, USA
rschulman@berkeley.edu
[2] California Institute of Technology, Pasadena, CA 91125, USA
winfree@caltech.edu

Abstract. Cairns-Smith has proposed that life began as structural patterns in clays that self-replicated during cycles of crystal growth and fragmentation. Complex, evolved crystal forms could then have catalyzed the formation of a more advanced genetic material. A crucial weakness of this theory is that it is unclear how complex crystals might arise through Darwinian selection. Here we investigate whether complex crystal patterns could evolve using a model system for crystal growth, DNA tile crystals, that is amenable to both theoretical and experimental inquiry. It was previously shown that in principle, the evolution of crystals assembled from a set of thousands of DNA tiles under very specific environmental conditions could produce arbitrarily complex patterns. Here we show that evolution driven only by the dearth of one monomer type could produce complex crystals from just 12 monomer types. The proposed mechanism of evolution is simple enough to test experimentally and is sufficiently general that it may apply to other DNA tile crystals or even to natural crystals, suggesting that complex crystals could evolve from simple starting materials because of relative differences in concentrations of the materials needed for growth.

1 Introduction

A plausible hypothesis for the origin of life on Earth must explain both *spontaneous self-replication*, i.e. how a self-replicating system first emerged, and *open-ended evolution*, i.e. how Darwinian evolution of this system led to complex organisms. Spontaneous self-replication requires that the components of the genetic material and the environment for replication existed on the early Earth, ideally in abundance, and that the assembly of these components into a replicator could reasonably have occurred. Open-ended evolution requires at minimum that complex genomes exist which would be fitter than all simpler genomes under plausible environmental conditions. A major difficulty in origin of life research is that while many hypotheses can explain the process by which either spontaneous self-replication or open-ended evolution might have occurred, no one hypothesis yet gives a detailed picture of the sequence of events leading to both.

There are many simple systems capable of replication that may have existed on the early Earth, including fire, autocatalytic reaction cycles [31] or lipid vesicles [32,30] for which it is not clear how Darwinian evolution might produce complexity. Similarly, there are complex systems, such as DNA combined with enzymes [21], ribozymes with specific oligonucleotide substrates [18], prions [17], DNA crystals [26], robots [16,14] or computer programs [1] for which complex evolution seems feasible, but how they might have arisen spontaneously is not clear.

Graham Cairns-Smith has proposed that polytypic clay crystals, which may have been common on the early Earth, could have been the first replicators [5,7]. If each layer within a clay crystal could contain monomers in any of several distinct arrangements (called layer types), then the sequence of layer types in a stack of layers would contain information. Crystal growth would propagate this information and crystals' occasional breaking would produce new growth fronts for its propagation, thereby amplifying it (Figure 1). Particular clay crystal sequences might promote catalytic functions that would be selected for, and eventually these crystal sequences would evolve into more complex sequences to promote more complex chemical functions, until the chemistry induced by the crystal's sequence became capable of self-replication on its own. Cairns-Smith called this scenario a "genetic takeover" [6].

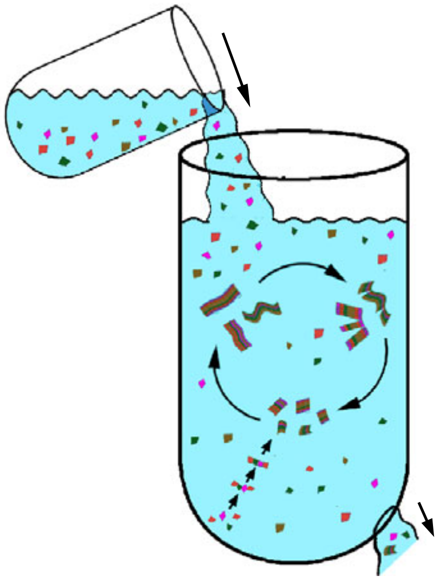


Fig. 1. Cairns-Smith's theory of crystal replication. Heterogeneous crystals propagate information consisting of the arrangement of monomers (shown as stripes) during growth. Crystal fragmentation creates new growth fronts to propagate the information. Monomers are replenished, and nucleation occasionally produces new crystals.

To date no one has demonstrated clay crystal information replication [4] and there is no direct experimental evidence that particular clay sequences have properties that might make them particularly efficient replicators [8]. However, one reason for its continued discussion [22] is that both prebiotic replication of clay patterns and the idea that some complex clay structures could have selective advantages seem reasonable. Since there has been only limited investigation into either of these questions, it is worth investigating whether clay might be capable of spontaneous self-replication and open-ended evolution.

However, it is unclear more generally how *any* complex crystal could arise through Darwinian selection. Here we consider whether there are features of crystal growth dynamics, rather than of chemistry particular to clay crystals, that might result in the evolution of complex crystals. We use DNA tile crystals, a model system for investigating generic features of 2 dimensional crystal growth [26]. DNA tile crystals can be readily studied experimentally (e.g. [38]) and theoretically using quantitative models (e.g. [37]).

One might be skeptical that generic features of crystal growth alone could be responsible for complex crystal evolution: It is generally assumed that a complex sequence would evolve because the sequence imbues some chemical functionality or otherwise alters the environment in a way that improves its reproductive fitness. However, a complex crystal may be selected for simply because it uses more abundant raw materials in its growth than simpler crystals. The preference for the addition of monomers that form multiple contacts with an existing crystal can put complex constraints on the order and frequency with which a crystal uses available monomers. It has been argued that these constraints could cause the selection of complex crystals [28], but so far only for crystals containing thousands of monomer types and under special physical conditions that are of limited relevance to natural crystal growth.

We will show that similar constraints could produce complex evolution in DNA tile crystals containing just 12 monomer types and over a wide set of physical conditions. We use a cellular automaton model [35] to enumerate the set of crystal morphologies that could be produced by a given set of monomers (a *tile set*). We show that if the monomer compositions of the crystal morphologies produced by a given tile set satisfy a simple property, then in a simplified model of growth we would predict selection of complex crystals. Using a combinatorial search, we find tile sets that satisfy this property. We then use a kinetic simulation to show that for at least one tile set, this evolution also occurs in a more realistic crystal growth model. While our search focuses on crystals containing 12 monomer types, it is reasonable to suppose that even smaller tile sets can exhibit similar phenomena.

2 DNA Tile Crystal Replication and Evolution

DNA tile crystals consist of tile monomers [13], rigid assemblies of oligonucleotides with short, single-stranded DNA segments, *sticky ends*, by which they bind to other tile monomers. Tiles can assemble into rectangular lattices to produce two-dimensional crystals [38] and ribbons [19,27,40]. The ease of working with DNA tiles make them a model system for crystal growth: DNA crystal monomers (tiles) are easily created by designing the tiles' oligonucleotide sequences, and the growth pathways of DNA crystals are relatively well-understood at the monomer level [24,9,27].

An example set of DNA tile monomers, i.e. a tile set, is shown in Figure 2(a). The diagram describes the possible interactions between the tile types, which determine the crystal growth dynamics. Edges with the same color that fit together like jigsaw puzzle pieces represent sticky ends with complementary sequences. The set consists of square *rule* tiles, and rectangular *top* and *bottom* edge tiles. These tiles can assemble to form a variety of patterned ribbon structures (Figure 2(d)). Close to the crystal melting temperature, the addition of a DNA tile to a crystal is energetically favorable only if it forms two or more bonds [36] (Figure 2(b)).

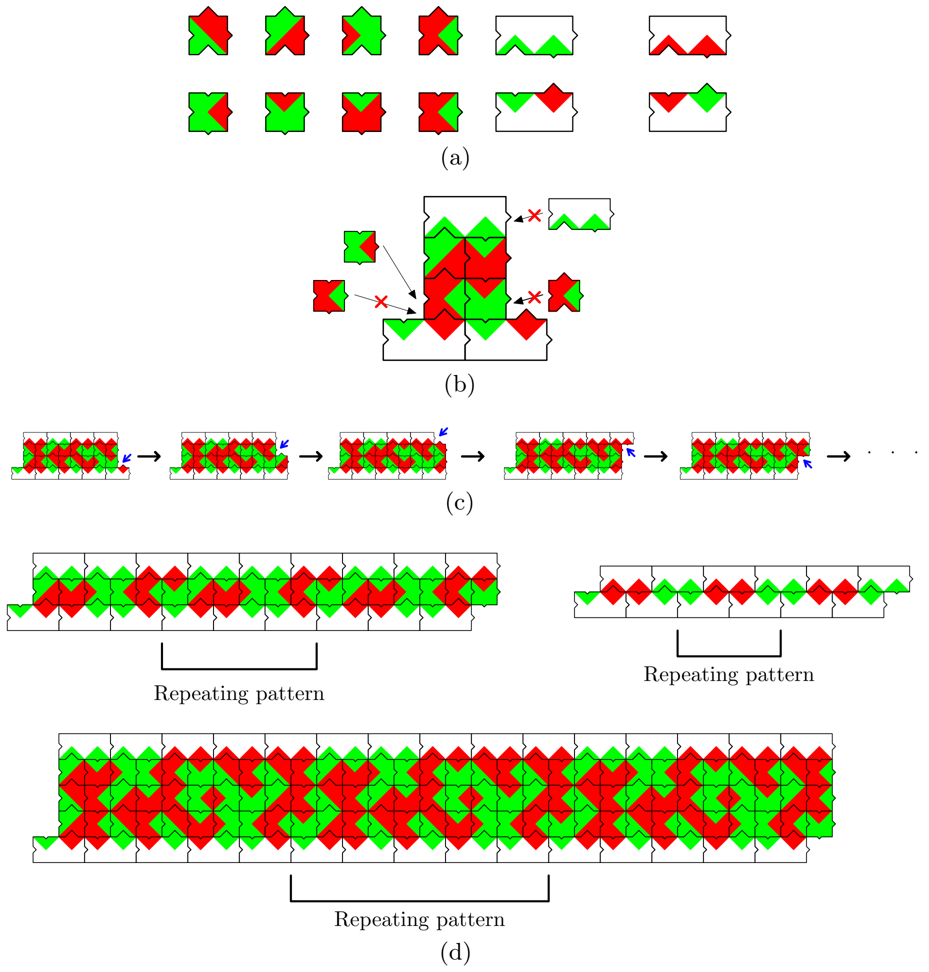


Fig. 2. An example zig-zag CA tile set. (a) The 12 tiles. Notches and colors represent the single-stranded sticky ends of each tile; interlocking edges of matching color have complementary sequences. Tiles cannot be rotated. **(b)** Close to the melting temperature tiles tend to attach to a crystal only where they match at least two edges. **(c)** Zig-zag crystal growth. At each step, a new tile may be added at the location designated by the small arrow. A simultaneous growth process also occurs on the crystal's left end. **(d)** Example assemblies of widths 2, 3 and 5 formed by the tiles in (a). The pattern repeated in each crystal is bracketed.

Under these conditions, for the tiles in Figure 2(a), a single column of tiles terminated by top and bottom edge tiles uniquely defines the preferred growth process and ribbon pattern (Figure 2(c)). Growth produces ribbons bearing repeating *wallpaper patterns* (Figure 2(d)). The wallpaper pattern particular to a crystal is its information, which would be replicated by the process Cairns-Smith proposed.

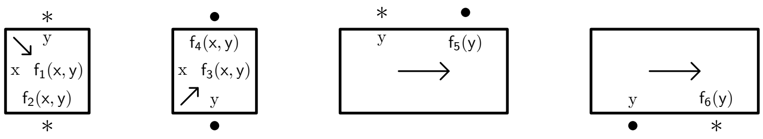
We are interested in whether evolution in this system might produce complex crystal forms. Crystal evolution occurs when (a) mistakes during crystal growth occur occasionally, producing mutations, and (b) some crystals replicate faster than others, i.e. are fitter. The fitness of a crystal is the geometric mean of crystal growth rate (in columns / time) and the per column rate of crystal fragmentation [28]. For a given tile set and crystal growth environment, we use mathematical models and simulations to estimate the growth and fragmentation rates of the crystals and thus their fitness.

A simplified version of the kTAM [25], a generalized crystal growth model applicable to DNA tile crystal growth [38,24,2] can be used to estimate crystal growth rates. We consider a version of this model in which (a) crystal growth proceeds exclusively by single tile addition (we ignore tile dissociation), (b) a tile may be added to a site if labels on at least two edges match those presented by the crystal at that site, and (c) monomer tiles arrive at potential binding sites with a frequency proportional to their concentration in solution. We assume that occasional violations of rule (b) produce mutations that introduce new crystal patterns into the population. Because we will show that growth rates can vary arbitrarily widely, we will for simplicity ignore the dependence of crystal width on fragmentation frequency and assume that fitness is proportional to crystal growth rate. However, under fluid shear, for example, breakage rates of crystals do decrease with crystal width [15].

Both the tile set, which is the alphabet that determines the types of ribbons that form, and physical conditions such as tile concentration determine the results of an evolutionary process. To ask whether crystal evolution could produce non-trivial genomes, we will examine the complexity of DNA tile crystals that are produced by an evolutionary process. For simplicity we measure complexity as a crystal's width. In Section 3 we develop a formalism that describes a family of tile sets we will consider. In Section 4 we then describe an environment, in terms of tile concentrations, where evolution of complex (wide) crystals could occur.

3 Binary Zig-Zag CA Tile Sets

To clearly enumerate the members of the family of tile sets we wish to consider as possible substrates for open-ended evolution, we define a cellular automaton variant, the *zig-zag cellular automaton* (or zig-zag CA). Computation on a zig-zag CA takes place on a lattice of defined width w and infinite length. The computation tape is a vertical column of width w, and the horizontal (lengthwise) axis of the lattice contains the computation history. Cell updates proceed alternately from the top edge to the bottom edge (downward), then from the



(a)

0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
$f_1(0,0)$	$f_1(0,1)$	$f_1(1,0)$	$f_1(1,1)$	$f_2(0,0)$	$f_2(0,1)$	$f_2(1,0)$	$f_2(1,1)$	$f_3(0,0)$	$f_3(0,1)$	$f_3(1,0)$	$f_3(1,1)$	$f_4(0,0)$	$f_4(0,1)$	$f_4(1,0)$	$f_4(1,1)$	$f_5(0)$	$f_5(1)$	$f_6(0)$	$f_6(1)$

(b)

Fig. 3. Abstract representation of zig-zag CA tiles. (a) Tiles with each set of inputs x and y compute as growth proceeds down the ribbon (left) and an analogous group of tiles compute as growth proceeds up (second to left). As denoted by the dots and stars, the input and output alphabets are not interchangeable between upward and downward computing tiles. The two top and two bottom tiles each have one input and one output. In this paper we consider the case where $f_1 ... f_6$ have Boolean inputs and outputs. In other figures, green and red represent 0 and 1. **(b)** The table used to generate a unique numerical identifier for a binary zig-zag CA tile set. The values of each output of the functions is written in the order shown give the identifier as a binary number, from most significant bit (left) to least (right).

bottom edge to the top edge (upward). The input to each cell consists of two values from a fixed alphabet $\mathcal{A}$: a right value and either a top or bottom value. During updates proceeding downward, the output right value is a function f_1 $(\{\mathcal{A},\mathcal{A}\} \rightarrow \mathcal{A})$ of the cell's input "right" value and of the cell above's input "bottom" value. The output bottom value is given by f_2 $(\{\mathcal{A},\mathcal{A}\} \rightarrow \mathcal{A})$, which takes the same inputs. Correspondingly, during upward updates, two functions f_3 and f_4 that operate on the cell's input "right" value and the cell below's input "top" value to determine the new right and top values respectively.

When a downward updating process reaches the bottom edge, the last bottom value is used to determine the edge top value that the bottom-most cell will use on the first upward update. A fifth function, f_5 $(\mathcal{A} \rightarrow \mathcal{A})$ determines the new top value given the edge bottom value of the previous row. Likewise, f_6 $(\mathcal{A} \rightarrow \mathcal{A})$, determines the new bottom value from the edge top value at the end of an upward series of updates. Figure 3(a) shows how to construct a zig-zag CA tile set from the functions $f_1 ... f_6$. For $|\mathcal{A}| = 2$, the set of tiles encodes a *binary zig-zag CA*.

Because computation takes place on a finite-width lattice, there are only a finite number of tape states that are possible: 2^{w-1} on a tape of width w ($w - 2$ cells have a right value and there is one top or bottom value). The tape's states, therefore, must repeat. If computation is logically irreversible, some states may be transient, i.e. on the path to a repeating cycle, but never themselves repeated.

(Details about the dynamics of finite cellular automata and illustrations of their state spaces, including cycles and transient states, are given in [39].) These repeating cycles are the wallpaper patterns in the zig-zag ribbon examples in Figure 2(d).

We will study the family of tile sets that implement binary zig-zag CA computation. Each tile set in the family contains 8 rule tiles (to encode the four possible binary inputs and their respective outputs in each of the up and down directions) and 4 edge tiles (two top tiles with the inputs 0 and 1 and their respective outputs and two bottom tiles with 0 and 1 inputs and their respective outputs), for a total of 12 tiles. Since a binary zig-zag CA tile set is defined by 4 two-input Boolean functions and 2 one-input Boolean functions (Figure 3(b)) and there are 16 two-input Boolean functions and 4 one-input Boolean functions, there are $16^4 \times 4^2 = 1048576$ binary zig-zag CA tile sets.

4 Predicting the Fitness of Zig-Zag CA Crystals

Growth of crystals is driven by diffusion of tiles into growth sites; the rate of tile attachment is thus proportional to tile concentration [20]. We consider a simple model of attachment rates where the time it takes a tile with concentration $[z]$ to attach is $\frac{1}{k_f[z]}$, where k_f is the forward rate constant of attachment [33], independent of tile type.

To examine how concentrations affect crystal growth rates and therefore crystal fitness, we consider the case where the rule tile concentrations are $[r]$, the concentrations of three of the four edge tiles (referred to as *common* edge tiles) are $[e]$, and the concentration of the fourth edge tile (referred to as the *rare* edge tile) is $[q]$. Let n be the number of rule tiles per column in the repeating pattern (cycle) of a crystal, c the number of common edge tiles used per cycle and u the number of rare edge tiles used per cycle. Since one edge tile is added per column, the average number of common edge tiles used per column is $\frac{c}{c+u}$ and the average number of rare edge tiles used per column is $\frac{u}{c+u}$. The average time to add a column of tiles, $\langle T \rangle$, is therefore:

$$\langle T \rangle = \frac{1}{k_f}\left(\frac{c}{(c+u)[e]} + \frac{u}{(c+u)[q]} + \frac{n}{[r]}\right). \tag{1}$$

When the concentration of all edge tiles is the same, i.e. $[q] = [e]$, $\langle T \rangle$ grows monotonically with the number of rule tiles in each column of the crystal. However, when $[q]$ is very small, the middle term dominates. A wider crystal with n_1 rows of rule tiles, c_1 common edge tiles, and u_1 rare edge tiles could grow faster than a thinner crystal with n_2, c_2, and u_2 rule, common edge and rare edge tiles, respectively, if $\langle T_1 \rangle < \langle T_2 \rangle$, i.e. the following equation is satisfied for $n_1 > n_2$:

$$(n_1 - n_2)\frac{1}{[r]} < \left(\frac{c_2}{c_2+u_2} - \frac{c_1}{c_1+u_1}\right)\frac{1}{[e]} + \left(\frac{u_2}{c_2+u_2} - \frac{u_1}{c_1+u_1}\right)\frac{1}{[q]}. \tag{2}$$

Thus when $\frac{1}{[q]} \gg \frac{1}{[e]}$, a wider crystal can grow more quickly than thinner ones if it uses particularly few rare edge tiles even though it must add more rule tiles per column. Further, for any case where $\frac{u_1}{c_1+u_1} < \frac{u_2}{c_2+u_2}$ it is possible to find an environment (in terms of $[r]$, $[e]$, and $[q]$) where the wider crystal grows more quickly.

5 Evolution of Binary Zig-Zag CA Crystals

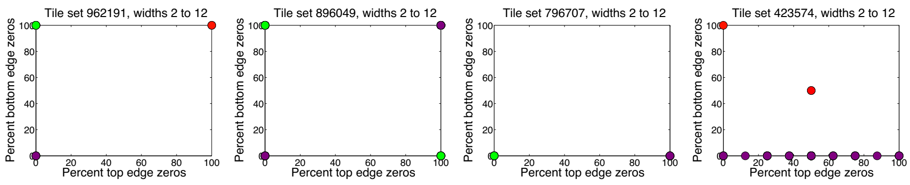
We sought to determine whether any binary zig-zag CA tile sets had the property that $\langle T \rangle$, the average time to add a crystal row, could decrease with width when one edge tile type was rare. For each tile set, we enumerated the patterns formed of particular widths. We then tabulated the frequency with which each pattern used top edge tiles with a "0" input vs top edge tiles with a "1" input, i.e. *top edge zeros* and *top edge ones*. Likewise, we tabulated the frequency that each pattern used *bottom edge zeros* and *bottom edge ones*.

As shown in Section 4, if the rare edge tile is present at a sufficiently low concentration, a pattern using few rare edge tiles per column is selected for. If a pattern of width w uses no rare edge tiles, no matter what the concentration of the rare edge tile, no pattern of width larger than w would be selected for simply because $[q]$ is low. In order to search for tile sets in which open-ended evolution might be feasible, therefore, we specifically searched for tile sets where no pattern could ever eliminate the rare edge tile, but where progressively wider patterns used the rare edge tile progressively less frequently than thinner patterns. We call these *evolvable tile sets*.

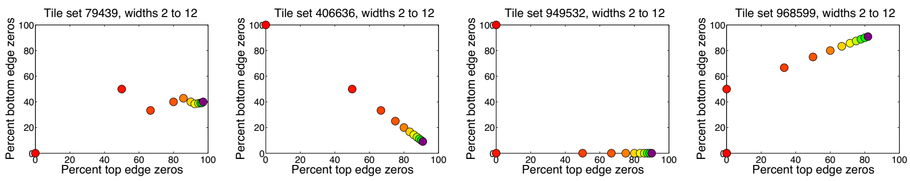
We first surveyed the edge tile usage of patterns of widths 2 to 12 of 4 randomly chosen tile sets (Figure 4(a)). In the first example (tile set 962191), all patterns either use only top edge zeros or top edge ones. Thus, if either tile type were rare, the thinnest type of crystal that used only the other top edge tile type would be the fittest. The same is true for bottom edge ones and zeros. No concentrations of edge tiles would be expected to induce the evolution of crystals wider than these crystals. The concentration of edge tiles used by the crystals in the other 3 examples follow the same sort of pattern. Since we did not observe that any of the 4 randomly chosen tile sets have the capacity for open-ended evolution as a result of a edge tile concentration differences, it is unlikely that most binary zig-zag CA tile sets are evolvable.

To find evolvable binary zig-zag CA tile sets, we surveyed all tile sets and identified those for which there was at least one edge tile that when rare, produced a fitter crystal with each increase in width from 2 through 7 tiles. This search produced 6144 putative evolvable tile sets. The edge tile usage of all patterns up to width 12 produced by a selection of these tile sets is shown in Figures 4(b-c).

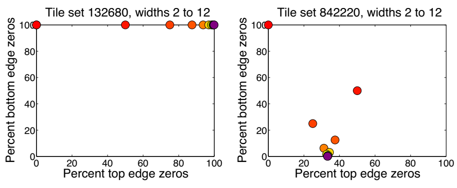
The rate at which rare tile type usage decreases with width determines the shape of the fitness landscape. In most putatively evolvable tile sets that we examined, the fittest pattern of a given width has a per-row usage of the rare edge tile of roughly either $\frac{1}{w}$ or $\frac{1}{2^w}$. In each case, it appears that one rare tile is used in each repeat unit of the fit pattern, and the repeat units have lengths either linear or exponential in the width.



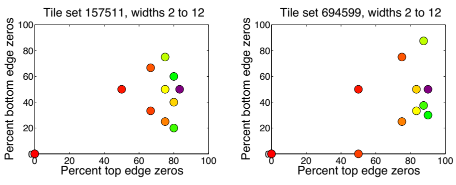
(a) Randomly selected tile sets



(b) Roughly $\frac{1}{w}$



(c) Roughly $\frac{1}{2^w}$



(d) Irregular

Fig. 4. Edge tile usage by zig-zag CA tile sets. The proportional usage of top and bottom zero and one tiles for each possible assembly of widths 2 to 12 for 12 representative tile sets. Each dot is one assembly type, and color indicates width, in rainbow order – red is width 2 and violet is width 12. The tile set number is computed as described in Figure 3(b) **(a)** 4 randomly selected tile sets. **(b-d)** Tile sets for which per-row usage of either or both top and bottom edge zero tiles decreases roughly as $\frac{1}{w}$ (b), $\frac{1}{2^w}$ (c), or irregularly (d) with width.

A second search of 100 random tile sets produced 7 tile sets for which fitter patterns appeared with some increases in width but not others. Figure 4(d) shows the usage patterns produced by two such tile sets. When the appropriate edge tile is rare, the first tile set produces a potentially fitter assembly type with assemblies of 1, 3, 5, 7, and 9 rows of rule tiles, and the second produces such new patterns with assemblies of 1, 2, 4, 5, and 10 tiles. Figure 5(c) shows some patterns produced by a rule of this type.

6 Evolution of Logically Reversible Zig-Zag CA Crystals

In the analysis in Sections 4 and 5, we assumed that the crystal's growth rate was the growth rate from its right end. However, in practice growth can occur from both the left and right ends of the crystals and the growth rate is the sum of the growth rates at each end.

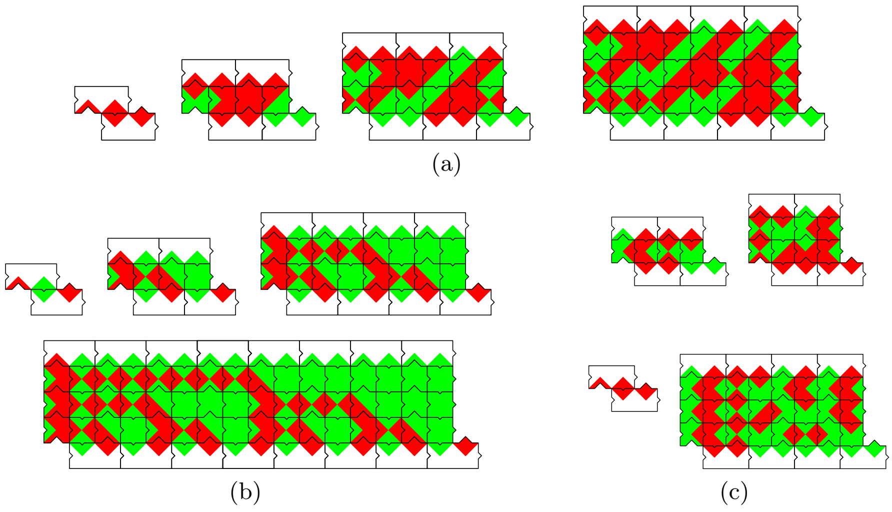


Fig. 5. Patterns of widths 2 to 5 of some representative evolvable tile sets. **(a)** Patterns from tile set 968599. **(b)** Patterns from tile set 132680. Note that the patterns produced by the tiles count in binary [11]. **(c)** Patterns from tile set 694599.

Growth from the right end is equivalent to running the tile set's zig-zag CA forward. By definition there is exactly one tile that can attach at the growth site by two edges simultaneously (i.e. energetically favorably) and the attachment of this tile creates a new growth site where exactly one tile can attach favorably (Figure 2(c)). The rightward growth rate is therefore easy to approximate by computing the rates of attachment of each tile in this series.

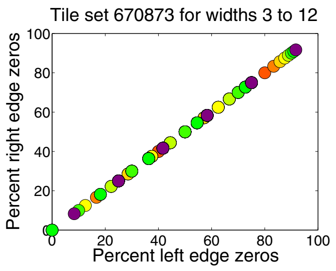


Fig. 6. Edge tile usage by a logically reversible zig-zag CA tile set. Format as in Figure 4.

Growth from the left end is equivalent to running the tile set's zig-zag CA backwards. Because a zig-zag CA is not guaranteed to be reversible, there may be no, one or multiple tiles that can fit at a given growth site. The dynamics of growth therefore can be more complex than in the rightward direction and we ignored these dynamics in Section 5. However, a subset of zig-zag CAs are logically reversible, i.e. there is always exactly one tile that can fit at the leftward growth site. A logically reversible zig-zag CA's left and right edges grow at the same speed. Our simple analysis of fitness landscapes based on growth rates is particularly accurate for these tile sets. Here we consider whether there are logically reversible zig-zag CA tile sets where a low concentration of an edge tile could drive evolution toward complex crystals, i.e. they are evolvable.

There are just 2304 types of logically reversible binary zig-zag cellular automata (an example being the tile set in Figure 2(a)). An exhaustive survey of these tile sets produced 16 for which each increase in width up to at least 12 could produce a new, fitter crystal (Figure 6).

For each of the 16 tile sets, the longest pattern of width w is $2w$ columns long, and with each increasing width the fittest pattern is the longest and uses just one rare edge tile.

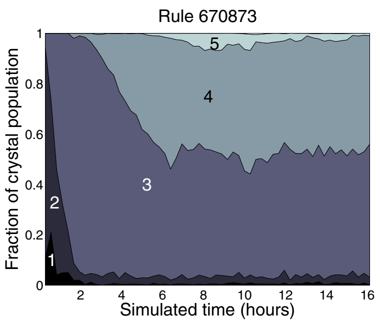
Because the cellular automaton is logically reversible, there can be no transient states; every column is included in a repeating pattern. All patterns are have a short repeat length, so the number of patterns increases exponentially with width. As a result, crystal growth is susceptible to growth errors: most mistakes will change the pattern being copied. This is in contrast to the evolvable logically irreversible tile sets that we found in Section 5, which had only one pattern type for each width.

7 Kinetic Simulation of an Evolution Process

To determine whether the proposed selection pressure results in wider ribbons in a more realistic crystal growth model, we simulated crystal growth and fragmentation using a full model of kinetic tile assembly (kTAM [36]) with the software package `xgrow` [34]. We compared simulated evolutionary dynamics for two tile sets. According to the simple analysis in Section 4, evolution of crystals assembled from the tile set we simulated (shown in Figure 2(a)) is predicted to produce wider crystals when either top or bottom edge ones were rare. The second tile set differed from the first only in that the outputs of the top 2 tiles were swapped. This small change also changed the fitness landscape such that the fittest crystal is predicted to have no rule tile rows.

In the simulation, tiles reversibly attached to each other or to existing assemblies with a diffusion dependent forward rate ($k_f = 10^6/M/s$ [33]) and a backward rate $k_r = k_f e^{-\Delta G^\circ/RT}$ set by the ΔG of tile attachment, which was assumed to be strictly cooperative: $\Delta G^\circ = -17$ kcal/mol for an attachment by two bonds and $\Delta G^\circ = -8.5$ kcal/mol for attachment by one bond. The energy for two sticky end bond attachments was chosen in order to be close to experimental measurements [27]. The concentration of free tiles was held constant, and crystal shearing was initiated at a given tile with rate $\frac{1}{2500}$ seconds with probability approximately $e^{-l/4}$ where l is the length of a vertically or horizontally oriented shearing path. Rule tiles, top edge tiles and the bottom edge zero tile were present at 2.25 μM, and the bottom edge one tile was present at 0.025 μM. The simulation tracked the crystals in a volume of 10^{-14} liters, and serial dilution was modelled by removing each crystal from the simulation with probability 0.5 when the mixture reached 1000 crystals consisting of at least 2 monomers.

Figure 7 shows the rate at which assemblies of different widths arise in the two simulations. An energetic barrier to nucleation [29] meant that more than 99% of crystals originated as fragments sheared from another crystal rather than



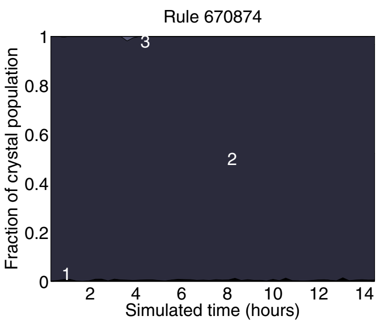


Fig. 7. Simulated evolution of zig-zag CA crystals. Numbers inside the graph indicate the width, including edge tiles. (a) Logically reversible tile set 670873, where the minimum possible usage of a rare edge tile increases with crystal width. (b) Logically reversible tile set 670874, where a crystal of width 2, which uses no rare edge tiles, is predicted to be fittest.

having spontaneously nucleated. The resulting dynamics show that wide crystals appear only for the tile set where this behavior was predicted by the analysis in Section 6. Equations 1 and 2 predict that for a fixed rare tile concentration, there should be an optimal width. Optimal widths for both tile sets in Figure 7 were observed.

8 Conclusions and Open Questions

This work suggests that the evolution of complex crystals from a simple set of monomers can be induced by differences in the monomer concentrations during crystal growth and replication. The components are simple in the sense that it would be feasible to synthesize their components and to watch their assembly and replication in the laboratory [3]. It will be important to determine how much the predictions made here hold up under a more realistic model of self-assembly. We must understand how growth errors and spontaneous nucleation of new crystals affect the evolutionary process. In any evolutionary process there is a bit-wise error rate (an error threshold) above which evolution becomes impossible [12]. There is reason to believe the tile sets that we study may be robust to many errors and therefore, that they can evolve even under imperfect assembly conditions. For example, in most of the logically irreversible cellular automata tile sets that we investigate, there is only 1 pattern that can be copied at a given crystal width. A mismatch error in such a tile set would simply change the segment of the pattern being copied at the growth front, rather than changing the pattern altogether.

Unfortunately, though, the highest acceptable bit-wise mutation rate and therefore the robustness of a sequence to mutation decreases with crystal width. Might we as a consequence expect a limit to the sizes of patterns that can be copied under attainable physical conditions? Naively, the answer to this question seems to be yes. But it may be that for some tile sets where the number of patterns a tile set can copy grows sub-exponentially with width, the error rate for the whole pattern may not increase with width.

However these questions are answered, this work already suggests two important points. First, simple crystals are capable of complex evolution. More investigation is needed to determine whether natural crystals are capable of complex evolution; specific crystals such as clays will often have fewer monomer types, more complex dynamics, and a greater variety of specific and nonspecific interactions between monomer types. Since many of the details of these dynamics and affinities are still unknown in clay and other natural crystal systems, simpler DNA tile systems such as DNA nanotubes [23,40] may also be useful in further investigations.

Second, our analysis suggests a more important point about complex evolution in simple systems. While much attention has been given to the chemical functionality of sequences, it is generally assumed that the dynamics of assembling a sequence are of ancillary interest in a Darwinian evolution process. In biology, where sequence information is often stored in linear polymers such as genomic DNA and all monomers bind to their sequence neighbors using the same chemistry, this may be mostly true. But the constraints in crystal patterns make logic a fertile ground for complex evolution, and similar constraints are likely for other kinds of chemical replicators. In considering the replication in a more general class of chemical systems, therefore, it is worth considering the contribution of logical evolution to the dynamics of an evolutionary process.

References

1. Adami, C.: Introduction to Artifical Life. Springer, Berlin (1998)
2. Barish, R.D., Rothemund, P.W.K., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano Letters 5, 2586–2592 (2005)
3. Barish, R.D., Schulman, R., Rothemund, P.W.K., Winfree, E.: An information-bearing seed for nucleating algorithmic self-assembly. Proceedings of the National Academy of Sciences USA 106(15), 6054–6059 (2009)
4. Bullard, T., Freudenthal, J., Avagyan, S., Kahr, B.: Test of Cairns-Smith's crystals-as-genes hypothesis. Faraday Discussions 136, 231–245 (2007)
5. Cairns-Smith, A.G.: The origin of life and the nature of the primitive gene. Journal of Theoretical Biology 10, 53–88 (1966)
6. Cairns-Smith, A.G.: Genetic Takeover and the Mineral Origins of Life. Cambridge University Press, Cambridge (1982)
7. Cairns-Smith, A.G.: The chemistry of materials for artificial Darwinian systems. International Reviews in Physical Chemistry 7, 209–250 (1988)
8. Cairns-Smith, A.G., Hartman, H.: Clay Minerals and the Origin of Life. Cambridge University Press, Cambridge (1986)
9. Chen, H.-L., Schulman, R., Goel, A., Winfree, E.: Reducing facet nucleation during algorithmic self-assembly. Nano Letters 7(9), 2912–2919 (2007)
10. Chen, J., Reif, J.H. (eds.): DNA 9. LNCS, vol. 2943. Springer, Heidelberg (2004)

11. Cook, M., Rothemund, P.W.K., Winfree, E.: Self-assembled circuit patterns. In: Chen and Reif [10], pp. 91–107
12. Eigen, M., McCaskill, J., Schuster, P.: Molecular quasi-species. Journal of Physical Chemistry 92, 6881–6891 (1988)
13. Fu, T.-J., Seeman, N.C.: DNA double-crossover molecules. Biochemistry 32, 3211–3220 (1993)
14. Griffith, S., Goldwater, D., Jacobson, J.M.: Self-replication from random parts. Nature 437, 636 (2005)
15. Hariadi, R.F., Yurke, B.: Elongational-flow-induced scission of DNA nanotubes in laminar flow. Physical Review E 82(4), 046307 (2010), http://pre.aps.org/abstract/PRE/v82/i4/e046307
16. Klavins, E.: Universal self-replication using graph grammars. In: 2004 International Conference on MEMS, NANO and Smart Systems (ICMENS 2004), pp. 198–204 (2004)
17. Li, J., Browning, S., Mahal, S.P., Oelschlegel, A.M., Weissmann, C.: Darwinian evolution of prions in cell culture. Science 327(5967), 869–872 (2010)
18. Lincoln, T.A., Joyce, G.F.: Self-sustained replication of an RNA enzyme. Science 323(5918), 1229–1232 (2009)
19. Mao, C., Sun, W., Seeman, N.C.: Designed two-dimensional DNA Holliday junction arrays visualized by atomic force microscopy. Journal of the American Chemical Society 121, 5437–5443 (1999)
20. Markov, I.V.: Crystal Growth for Beginners. World Scientific, Singapore (2003)
21. Mills, D., Peterson, N., Spiegelman, S.: An extracellular Darwinian experiment with a self-duplicating nucleic acid molecule. Proceedings of the National Academy of Sciences USA 58, 217–224 (1967)
22. Orgel, L.E., Crick, F.H.C.: Anticipating an RNA world. Some past speculations on the origin of life: Where are they today? FASEB Journal 7, 238–239 (1993)
23. Rothemund, P.W.K., Ekani-Nkodo, A., Papadakis, N., Kumar, A., Fygenson, D.K., Winfree, E.: Design and characterization of programmable DNA nanotubes. Journal of the American Chemical Society 126(50), 16344–16352 (2004)
24. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLOS Biology 2, 424–436 (2004)
25. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares. In: Symposium on Theory of Computing (STOC), pp. 459–468. ACM, New York (2000)
26. Schulman, R., Winfree, E.: Self-replication and evolution of DNA crystals. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS (LNAI), vol. 3630, pp. 734–743. Springer, Heidelberg (2005)
27. Schulman, R., Winfree, E.: Synthesis of crystals with a programmable kinetic barrier to nucleation. Proceedings of the National Academy of Sciences USA 104(39), 15236–15241 (2007)
28. Schulman, R., Winfree, E.: How crystals that sense and respond to their environments could evolve. Natural Computing 7, 219–237 (2008)
29. Schulman, R., Winfree, E.: Programmable control of nucleation for algorithmic self-assembly. SIAM Journal on Computation 39, 1581–1616 (2009)
30. Segré, D., Ben-Eli, D., Deamer, D.W., Lancet, D.: The lipid world. Origins of Life and Evolution of Biospheres 31(1-2), 119–145 (2001)
31. Wächtersäuser, G.: Before enzymes and templates: theory of surface metabolism. Microbiology and Molecular Biology Reviews 52(4), 452–484 (1988)

32. Walde, P., Wick, R., Fresta, M., Mangone, A., Luisi, P.L.: Autopoetic self-reproduction of fatty acid vesicles. Journal of the American Chemical Society 116, 11649–11654 (1994)
33. Wetmur, J.G., Fresco, J.: DNA probes: Applications of the principles of nucleic acid hybridization. Critical Reviews in Biochemistry and Molecular Biology 26(3-4), 227–259 (1991)
34. Winfree, E.: The xgrow simulator, http://www.dna.caltech.edu/Xgrow/
35. Winfree, E.: On the computational power of DNA annealing and ligation. In: Lipton, R.J., Baum, E.B. (eds.) DNA Based Computers. DIMACS, vol. 27, pp. 199–221. American Mathematical Society, Providence (1996)
36. Winfree, E.: Simulations of computing by self-assembly. Technical Report CS-TR:1998.22, Caltech (1998)
37. Winfree, E., Bekbolatov, R.: Proofreading tile sets: Error-correction for algorithmic self-assembly. In: Chen and Reif [10], pp. 126–144
38. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539–544 (1998)
39. Wuensche, A., Lesser, M.: The Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata. Perseus Books, Cambridge (1992)
40. Yin, P., Hariadi, R.F., Sahu, S., Choi, H.M.T., Park, S.H., LaBean, T.H., Reif, J.H.: Programming DNA tube circumferences. Science 321, 824–826 (2008)

Towards Domain-Based Sequence Design for DNA Strand Displacement Reactions

David Yu Zhang

California Institute of Technology, Pasadena, CA, USA
dzhang@dna.caltech.edu

Abstract. DNA strand displacement has been used to construct a variety of components, devices, and circuits. The sequences of involved nucleic acid molecules can greatly influence the kinetics and function of strand displacement reactions. To facilitate consideration of spurious reactions during the design process, one common strategy is to subdivide DNA strands into domains, continuous nucleic acid bases that can be abstracted to act as a unit in hybridization and dissociation. Here, considerations for domain-based sequence design are discussed, and heuristics are presented for the sequence design of domains. Based on these heuristics, a randomized algorithm is implemented for sequence design.

1 Introduction

DNA strand displacement is a process through which a single-stranded DNA molecule (*strand*) reacts with a multi-stranded DNA *complex* to release another DNA strand (Fig. 1AB). Typically, strand displacement is facilitated by *toeholds*, short complementary single-stranded domains that act to colocalize the invading strand with the complex. The thermodynamics of the toehold region can be calculated based on sequence [1] [2] [3], and largely determine the kinetics of the strand displacement reaction if the invading strand does not possess significant secondary structure [4].

DNA strand displacement has been used to construct a number of dynamic DNA devices, including logic gates and circuits [5] [7] [6] [8] [9], catalytic reactions and networks [11] [10] [12] [13] [14] [15] [16], and nanoscale motors and walkers [17] [18] [19] [20]. These devices' mechanisms are based on Watson-Crick complementarity, and are expected to function for a wide variety of DNA sequences. Nevertheless, the kinetics of these devices' function are slowed if the sequences possess significant secondary structure (Fig. 1C) [21] [22]. As strand displacement-based DNA devices become more reliable, many different such devices will be integrated to construct complex reaction networks with more advanced functions. Many different DNA molecules must thus perform their function simultaneously without interfering with each other. Thus, an automated method for DNA sequence design is needed for strand displacement-based DNA devices.

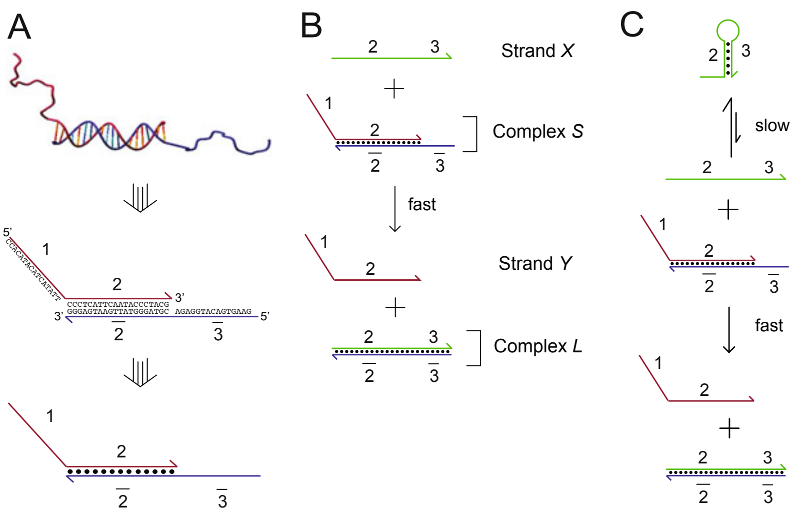


Fig. 1. DNA strand displacement **(A)** DNA abstraction. DNA strands and complexes are represented by directional lines, with the hook denoting the 3' end. DNA strands can be functionally abstracted into domains, consecutive bases of DNA that act as a unit in binding and dissociation. Domains are represented by numbers; a barred domain denotes a domain complementary in sequence to the unbarred domain (e.g. domain $\bar{2}$ is complementary to domain 2). Domain 2 is referred to in this paper as a "branch migration domain" and domain $\bar{2}$ is referred to as a "complement domain." Domains 3 and $\bar{3}$ are referred to as "toehold domains." **(B)** DNA strand displacement. **(C)** Secondary structure could hinder the kinetics of strand displacement. Strand X needs to unfold into a free state before it can branch migrate.

Although excellent algorithmic methods for DNA sequence design have been presented [23] [24] [25] [26] [27] [28], these methods generally maximize the probability of DNA strands forming desired structures and complexes at equilibrium, such as in the case for many DNA self assembly applications. For strand displacement-based devices and networks, thermodynamics-based methods of sequence design are not guaranteed to provide satisfactory sequences, because they neglect the consideration of kinetic pathways between DNA strand and complex states. In Fig. 2, for example, the strand shown in Fig. 2A may proceed through strand displacement slower than the strand in Fig. 2B, despite being the former possessing a minimum free energy (mfe) structure with standard free energy (ΔG°) closer to 0.

This paper presents a domain-based approach to sequence design of strand displacement-based reaction, networks, and devices. Domains are consecutive bases that serve as functional units in binding and dissociation (Fig. 1A), providing a useful abstraction for the design DNA devices. The sequences of involved DNA strands are obtained by concatenating the sequences of the strand's constituent domains. This domain-based approach to sequence design is simple and generalizes to many different possible DNA complexes and reactions, but is limited in its ability to eliminate undesirable spurious hybridization between DNA strands, particularly at the interface between different domains.

The other extreme, sequence design based on thermodynamic and kinetic analysis of all possible intermediates in strand displacement reactions, would potentially allow a rigorous method for generating optimal sequences for any reaction network. However, such a sequence designer would require significantly more computational resources, and is not available at present time. Additionally, although the thermodynamics of most DNA structural motifs have been carefully characterized over the past 20 years [1] [2] [3], two particular ones relevant to the analysis of multi-stranded intermediate complexes remain elusive: the energetics of pseudo-knotted complexes [29] [30] [31] and coaxial stacks [32] [33] [34] [35]. This incompleteness of DNA thermodynamic data suggest that even these more complex methods may not generate truly “optimal” sequences.

2 Considerations

In this section, considerations for domain-based sequence design are presented. These considerations are considered generally relevant to sequence design for DNA constructions involving strand displacement.

Avoidance of Long Continuous Regions of Spurious Hybridization. Long continuous regions of spurious binding in the branch migration domain should be avoided with priority over several shorter regions of spurious binding, even if the latter result in a more negative minimum free energy structure for the domain or strand. The reason for this is because branch migration is a sequential process: the binding energy of the strongest continuous region of hybridization likely determines the activation energy of the branch migration process.

When multiple different helices are present as in Fig. 2B, only the first of them needs to spontaneously open in order for branch migration to initiate. In contrast, when long continuous regions of spurious hybridization exist, branch migration over each of these structured bases is energetically unfavorable, and the kinetics of the overall branch migration process will be slowed exponentially in the energy of the binding region. To take a numerical example, if the hairpin shown in Fig. 2A has energy of -10 kcal/mol, while each of the hairpins in Fig. 2B has energy -7 kcal/mol, the kinetics of the strand displacement reaction may be a factor of 100 faster for reaction in Fig. 2B at room temperature, despite having an strand mfe of -14 kcal/mol.

Furthermore, because the displaced domain is the same sequence as the branch migration domain, the displaced portion of the domain will start forming the structures that spontaneously opened in branch migration domain. For example, in the bottom-left panel of Fig. 2B, the red strand forms the hairpin on its 3’ end, analogous to the 3’ hairpin of the blue strand. The formation of secondary structure by the displaced strand thus can facilitate branch migration.

There are other possible mechanisms for branch migration through highly structured domains of DNA, such as through 4-way branch migration. However,

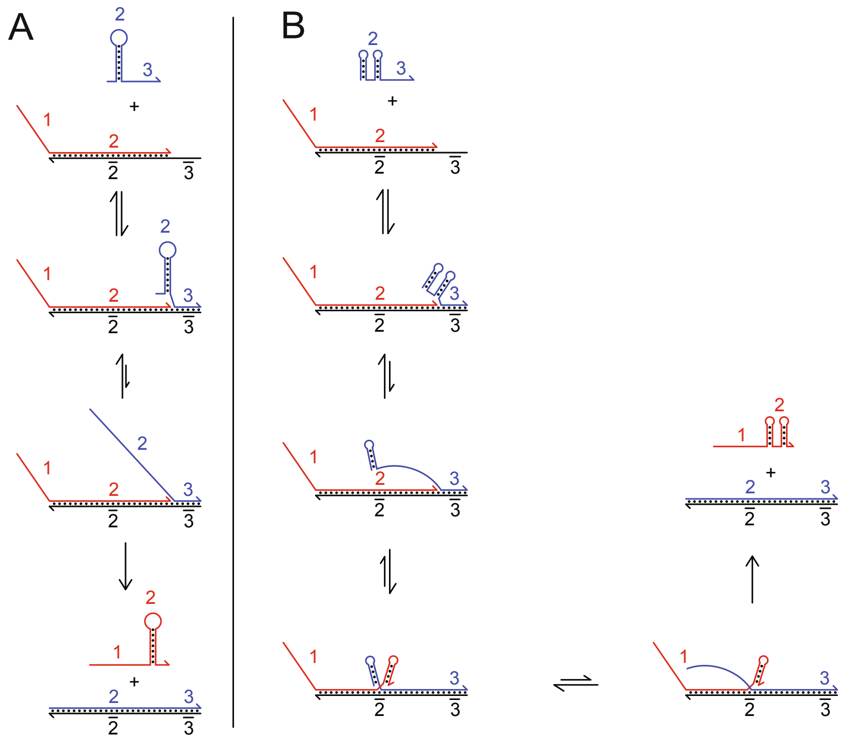


Fig. 2. Failure of thermodynamics-based sequence design. The ΔG° (standard free energy) of the folded domain 2 is more negative in **(B)** than in **(A)**, but the sequential nature of branch migration means that the strand displacement reaction involving **(B)** could be faster than that of **(A)**.

the author expects that most sequences generated with intent to avoid spurious hybridization will exhibit low enough amounts of spurious hybridization that the dominant pathway for branch migration will be through spontaneous dissociation of spurious hybridization.

Interaction and Crosstalk: Domain Concentration Effects. We define a "branch migration domain" to be a domain which competes to binding to a "complement domain" via branch migration. In Fig. 1, the 2 domain is a branch migration domain, while the $\bar{2}$ domain is a complement domain. For consistency, branch migration domains are all represented by unbarred numbers.

In strand displacement reactions, there is a necessary excess of branch migration domains over complement domains. For example, there are two copies of domain 2 in Fig. 2, but only one copy of domain $\bar{2}$. This excess manifests as a very large difference in the concentrations of single-stranded branch migration domains and complement domains. Almost all of the complement domains will be double-stranded at all times, while the branch migration domains will be single-stranded in significant concentrations.

Spurious hybridization occurs only between two single-stranded domains; this causes potential spurious hybridization involving a complement domain to be significantly less likely, and that between two complement domains to be nearly non-existent. Consequently, sequence design should primarily seek to minimize spurious binding between different branch migration domains (here refered to as *interactions*, Fig. 3B). Spurious hybridization between branch migration domains and complement domains (here refered to as *crosstalk*, Fig. 3B) is also undesirable insofar as two different domains may non-specifically displace each other in binding to their respective complements. However, mismatches destabilize the thermodynamics and also significantly impede the kinetics of branch migration [13] [36]. Because most domain sequences designed will differ from each other by at least a few bases, crosstalk is usually not a problem in practice.

Of interactions, self-interactions (wherein a domain significantly hybridizes to an identical copy of itself) can be considered the most problematic for two reasons: First, intra-domain and intra-molecular hybridization are entropically favored because of high local concentration. Second, assuming the thermodynamics of all interactions to be equal, dimerization is likely to be more prevalent than other interactions, because the concentration of a single-stranded domain correlates perfectly with itself (while different domains may not be single-stranded in high concentration at the same time in dynamic circuits).

Thus, sequence design for branch migration domains should avoid domain sequences with self-interactions with highest priority, other interactions with secondary priority, and minimize crosstalk only insofar as it does not hurt the previous two criteria. For sequence design of toehold domains, crosstalk is a larger consideration on the par of interactions.

Minimizing Guanine Frequency in Branch Migration Domains. Of the four canonical DNA nucleotides, guanine (G) stands out in being the most problematic with regards to sequence design for synthetic biology purposes, both because it is promiscuous (binding to thymine (T) nearly as strongly as adenine (A) [1]) and because it can form guanine quartets [37], a quadruplex structure based on non-Watson Crick binding, usually requiring at least 4 consecutive G's.

For these reasons, it is desirable to minimize the total concentration of guanine nucleotides used in the system. As pointed out previously, branch migration domains are necessarily in higher concentrations than complement domains–this leads to a natural strategy of minimizing the frequency of G's in branch migration domains. This strategy is further desirable because both the branch migration domains and the complement domains are now constructed with only 3 of the 4 nucleotides (C/A/T for the former, G/A/T for the latter), which drastically reduces the chance of self-interactions.

The idea of using only some of the four nucleotide bases in designing DNA sequences was first proposed by Mir [39], who suggested that restricted alphabets may practically avoid nonspecific hybridization, based on experimentally observed hybridization behavior of various sequences [38]. Experimental use of DNA strands with restricted alphabets for DNA nanotechnology purposes is relatively recent [10] [4] [40].

Avoidance of Long A/T and Long G/C Regions. For certain applications, it may be desirable to avoid long uninterrupted stretches of weak (A/T) bases or strong (G/C) bases. Long continuous regions of weak A/T bindings will melt at significantly lower temperatures and breathe significantly more than those with more mixed base distributions. Additionally, the hybridization rate constant of domains with only A/T bases have been reported to be an order of magnitude lower than that of one with a uniform distribution of G/C/A/T bases [4].

On the other hand, long continuous regions of strong G/C bindings are more likely to be spuriously hybridized to other strands and domains, because the strong G/C binding will counteract the destabilizing influences of DNA bulges and mismatches [1]. Additionally, long continuous regions of C/G sequence are more likely to adopt Z-DNA configurations [41]. Finally, branch migrate are likely to proceed more slowly in G/C rich regions due to their stronger base stacking thermodynamics, which may in turn necessitate stronger toehold domains for fast strand displacement [4].

Breathing Near the End of Helices. One frequent concern in the design and construction of strand displacement reactions and networks is blunt end strand exchange, strand displacement in the absence of single-stranded toehold domains. The rate constant of blunt end strand exchange has been reported to be on the order of 1 M^{-1} s^{-1} for multiple sequences [42] [43] [4]. This rate constant could be significantly higher if the branch migration domain is terminated with A or T.

The mechanism for blunt end strand exchange is postulated as the following: The base pairs at the end of a double-stranded branch migration domain "breathes," temporarily unbinding despite the bound state being more favorable. Any strands possessing the branch migration domain that happens to be in the local vicinity effectively has a few bases of toehold until the ends of the complex re-hybridizes. Terminating branch migration domains with A/T base pairs thus is likely to increase the rate constant of blunt end strand exchange because A/T bases are weaker than G/C ones. Furthermore, measured thermodynamic parameters show that DNA helices are destabilized when closed by an A-T base pair (by 0.15 kcal/mol at room temperature) [1], in addition to the weaker binding thermodynamics.

Consequently, it is desirable for all branch migration and complement domains to start and end with G or C nucleotides. For reasons given previously, it is recommended that branch migration domains start and end with C's.

Interface Between Domains. One problem specific to domain-based sequence design is the interface between domains. Typically, a stretch of binding typically requires 3-4 consecutive complementary bases in order to be stable, and such binding will be not be visible to the sequence design software if the bases span multiple domains. Rather than abandoning domain-based sequence design altogether, it should be possible to weight spurious hybridization near the ends of the domains appropriately so that concatenated domains are unlikely to form long spurious hybridization regions at their interface.

Consideration of the potential interface problems through calcuating the interactions and crosstalks for all pairwise concatenations of domains is not recommended for two reasons: First, this causes a runtime slowdown quartic in the number of domains (N^4 pair-wise folds needed for N^2 pair-wise concatenations, where N is the original number of domains), negating the speed advantage of low-level domain-based design software. Second, many of the possible pairwise concatenations will not actually be present in solution, and optimizing the interaction and crosstalk potential of these non-existent domain combinations will actually worsen the crosstalk and interaction problems of the strands that do exist.

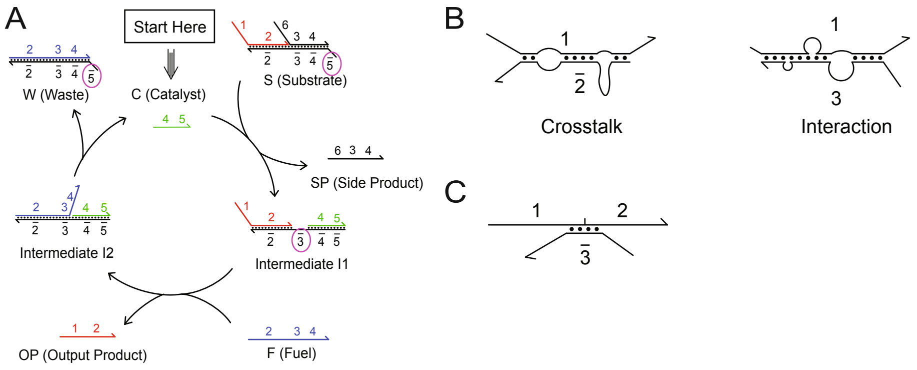


Fig. 3. Spurious partial hybridization between unrelated domains. **(A)** Asymmetry of single-stranded prevalence. In the catalytic reaction cycle (adapted from Zhang et al. [10]), the only barred domains that appear in single-stranded forms are $\bar{3}$ and $\bar{5}$, the toeholds. **(B)** Accordingly, we draw a distinction between "crosstalk" and "interaction," with the former denoting spurious partial hybridization between an unbarred domain and a barred domain, while the latter refers to the hybridization between two unbarred domains. Interactions are likely to have greater effects on kinetics than crosstalk. **(C)** One shortcoming of domain-based sequence design are the possibility of crosstalk and/or interactions at the interface between two domains.

3 Implementation

The Domain Design (DD) software presented here employs heuristics to quantify the considerations discussed in the previous section; these heuristics are used to generate an overall "score" for each domain, the worst (maximum) of which is the global score for a set of domains. Simply put, the score is a metric that roughly correlates with the likelihood that the kinetics of strand displacement reactions deviate from predictable models [4]. The overall score for a set of domains is simply the maximum (worst) of the domain scores. DD's algorithm performs sequence design by starting with a random set of sequences, and continually attempting to mutate these sequences to achieve improved (lower) scores.

DD assumes that the domains designed are branch migration domains or the unbarred toehold domains, and evaluates crosstalk potential by automatically generating the complements to all designed domains. One good design strategy may be to design all the toehold domains first, and subsequently design the branch migration domains (with the toehold domain sequences locked–see User Interactivity: Base locking).

The algorithm uses a number of scoring parameters, some of which are hard-coded and can be turned on or off at the user's discretion. The values of the hard-coded parameters are by no means guaranteed to be optimal or even close to optimal–these represent only the author's best guess at good parameter values for designing 10 or fewer different domains, each of length between 5 and 30 nucleotides. The pseudocode for the Domain Design software is given in Fig. 4.

Score Calculation. The score of a domain is computed as the sum of the domain intrinsic score and the worst of its crosstalks and interactions.

The domain intrinsic score accounts for score penalties based only on the sequence of the domain, rather than potential spurious hybridization. Domains with four consecutive G's or C's receive a +50 to score, so as to strongly discourage the formation of G quartets. Domains with six consecutive G/C nucleotides or six consecutive A/T nucleotides receive a +20 to score, if the user elects to turn on the option for avoiding long regions of G/C binding and long regions of A/T binding. Finally, the user-defined "importance" of the domain is also added to the score (see User Interactivity: Domain importance).

The crosstalk and interaction scores are evaluated similarly, by evaluating every potential way that the two domains can spurious hybridize and identifying the way that yields the highest score. The score of each way of binding is calculated as thus: each G-C base pair contribute +2 score and each A-T pair contribute +1 score. Base pairing of x consecutive nucleotides without intervening bulges or mismatches gain a further score of $+2^{x-4}$ for $x \geq 5$, thus strongly discouraging the formation of long unbroken stretches for spurious hybridization. Mismatch and bulge structures between stretches of hybridization each contribute -3 score, with an additional -0.5 score for every base in the bulge or mismatch in excess of 1. The scores of hybridization segments starting at the first base of a domain or ending with the last base of a domain are increased by +3, to reduce the potential for interface problems when concatenating domains.

The difference in DD's treatment of crosstalk, interactions, and self-interactions is that each's score is modified linearly. Self-interactions receive a +5 to score, interactions default to no adjustment, and crosstalk scores are divided by 2 and then a -10 to score is applied. Thus, DD places higher priority on self-interactions and lower priority on crosstalk.

In practice, DD will optimize the sequences of a set of 10 domains, each 20 nt long, in about a minute to the point where the overall score of the system is between +10 and +15.

Algorithm. DD calculates interaction and crosstalk score of a pair of domains by using a dynamic programming algorithm similar to those used mFold, DINAMelt, and PairFold [44] [3] [45]. See source code for details.

In every run of the main loop, DD attempts to improve the overall score by mutating one of the domains. If the overall score was improved through the mutation, DD keeps the mutation. If the overall score was unchanged through the mutation, DD keeps the mutation with 0.2 probability. If the overall score was worsened through the mutation, DD discards the mutations. At the user's option, DD will either randomly select one of the domains for attempted mutation with uniform probability, or will target the domain currently with the worst score with probability $\frac{1}{3} + \frac{2}{3N}$, where N is the number of domains designed. Targeting the domain with worst score for mutations is expected to improve the speed of the software.

The number of bases attempted to be mutated is roughly exponentially distributed, with $\frac{1}{2}$ probability of mutating 1 base, $\frac{1}{4}$ of mutating 2 bases, etc., up to 10 bases or M (the number of bases in the domain), which ever is smaller. The simultaneous mutation of multiple bases is thought to prevent the domain sequences from entering local score optima.

Each base attempted to be mutated is replaced by a random base drawn from the bases allowed to occur within the domain (see User Interactivity: Base Composition). This means that there is some probability that an attempted mutation does not actually change the base, because the new base happens to be the same as the old. At the user's option, the mutation process can be biased against the incorporation of G, so that only 4% of attempted mutations result in a G (assuming G is an allowed base in the first place).

Time and Space Complexity. The time complexity of the algorithm can be estimated easily. Define the problem to be the design of N different domains, each of length M. After every mutation attempt, the software must update the interaction and crosstalk scores of all domains with the mutated domain, which involves checking $\mathrm{O}(N)$ interactions and crosstalks. Calculating an interaction or crosstalk score using dynamic programming requires $\mathrm{O}(M^2)$ time. Thus, the algorithm has time complexity $\mathrm{O}(NM^2)$ per mutation attempt.

The number of total mutation attempts needed to in order to grant each domain a fixed expected number of mutations attempts will scale linearly with N. Thus, DD requires $\mathrm{O}(N^2M^2)$ time to reduce the overall score to decent levels.

In addition to the $\mathrm{O}(NM)$ space needed to store the sequences of the domains, there are two major contributions to the space complexity of the algorithm. First, during the evaluation of a crosstalk or interaction score, $\mathrm{O}(M^2)$ space is needed for the dynamic programming algorithm. Second, there are $\mathrm{O}(N^2)$ pairwise crosstalk and interaction scores between the N domains; these scores need to be stored in order to allow fast score updating (not storing the scores worsens the time complexity by a factor of N). Thus, the overall space complexity of the algorithm is $\mathrm{O}(N^2 + NM + M^2) = \mathrm{O}(N^2 + M^2)$.

4 User Interactivity

The Domain Design software (DD) presented here was written with the intention of being a tool for helping the informed DNA engineer design sequences, rather

```
FOR i = 1:N
  FOR j = 1:N
    Score(i,j) = Domain_intrinsic_score(i) + Max(Crosstalk(i,j), Interaction(i,j))
  END FOR
  Domain_Score(i) = Max_over_j(Score(i,j))
END FOR
Global_score = Max_over_i(Domain_Score(i))

WHILE (TRUE)
  k = Random(1,N)
  New_domain(k) = Mutate(Domain(k))

  FOR i = 1:M
    New_Score(i, k) = Domain_intrinsic_score(i) + Max(Crosstalk(i,k), Interaction(i,k))
    New_Score(k, i) = New_Score(i,k)
    IF (i != k)
      Domain_Score(i) = Max_over_j(Score(i,j))
    END IF
  END FOR
  Domain_Score(k) = Max_over_j(Score(k, j)
  New_global_score = Max_over_i(Domain_Score(i))

  IF (New_global_score < Global_Score)
    Domain(k) = New_domain(k)
    FOR i = 1:M
      Score(i,k) = New_score(i,k)
      Score(k,i) = New_score(k,i)
    END FOR
    Global_Score = New_global_score
  END IF

  IF (Keyboard_input())
    Prompt_user_menu()
  END IF
END WHILE
```

Fig. 4. Pseudo-code for the Domain Design software

than being an arbitrator of sequences. Accordingly, Domain Design strives to maximize the ease with which the user can modify and adjust the sequence design process: The user may pause the sequence optimization process at any time to tweak sequences as well as a number of design parameters, including the ones listed below. User interactive sequence design is considered advantageous because the user may be able to identify sequence problems that the software is not able to detect.

Base Locking. DD acknowledges that the user may possess certain constraints or preferences in the design of sequences. For example, the user may require the promoter sequence for the T7 RNA polymerase [46] [47] or a deoxyribozyme sequence [48] [49] [50] to be present at a given location. To this end, the user can manually change the sequences of any domain, and can furthermore lock part or all of a domain sequence from being mutated by the software. Locked bases are visually displayed in red (Fig. 5A).

Domain Importance. Different domains serve different purposes in a reaction network, and the frequency of a domain being single-stranded also varies among domains. For example, toehold domains are required to colocalize the reactants of the strand displacement reaction, and interactions/crosstalk involving toeholds may slow the kinetics of strand displacement much more than similar interactions/crosstalk between branch migration domains. Consequently, sequence-based interactions and crosstalk in the different domains need to be weighed differently. A rigorous and justified method for weighing the domains must take into account the the context of the overall reaction network, and is beyond the scope of a low-level domain-based approach to sequence design.

Instead, DD implements a user-defined "importance" parameter, which acts as an additive term to the score of the domain (Fig. 5B). In DD, the overall score of the set of domain sequences is determined by the domain with the worst (highest) score, and in a typical run the scores of all of the domains will be similar. The additive importance term to score means that high importance domains must have lower values for the other score terms in order to have similar score to domains with low importance.

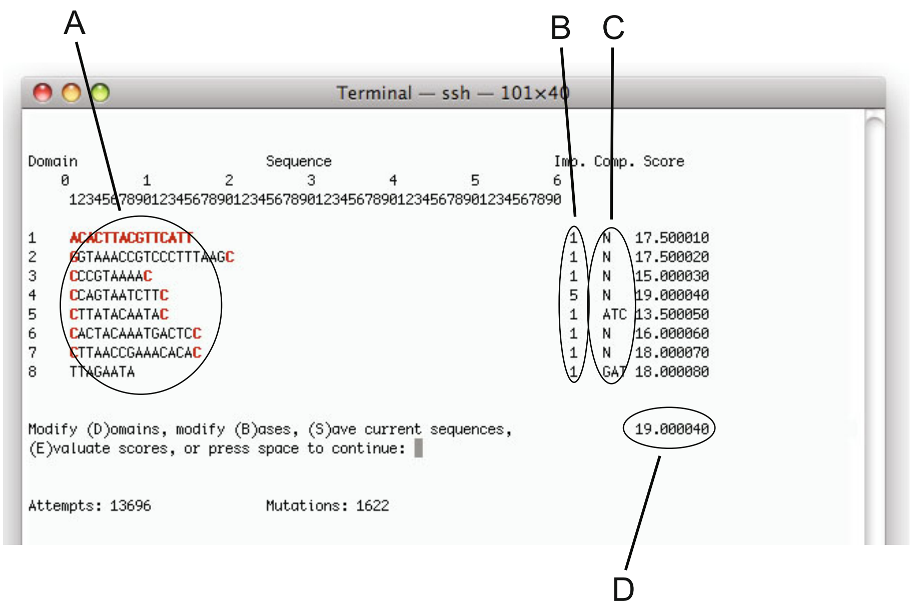
Base Composition. DD allows the user to specify the base compositions of each of the domains. For reasons outlined previously, it may be desirable to design certain domains using 3-letter C/A/T alphabets. Furthermore, the user also may wish to restrict certain domains to only A/T bases or G/C bases, so that the domain is weak- or strong-binding. This feature also facilitates the sequence design of non-standard DNA structures, such as Z-DNA [41] (C/G), or Hoogsteen triplex bindings (C/T) [51].

Designing Sequences to Function with an Existing System. Finite research funding usually implies that DNA nanotechnologists will tend to use and reuse the same DNA oligonucleotide strands for many different purposes. It is not only economical but also practical, because using previously tested strands eliminates the chance for unexpected sequence-specific problems. Given this tendency, it is therefore important for sequence design software to be able to design optimal sequences given the constraints of the sequences of the existing strands.

DD allows the user to load sequences from files before attempting to design new sequences. At the user's option, the domains loaded in this manner can be all be locked for the new design process. The user may also save the sequences generated by DD at any time, and the save file will include information such as the importance, composition, and lock status of all bases and domains.

5 Discussion

This paper discusses the author's considerations for sequence design of DNA strand displacement-based reactions and cascades, and makes a tentative relative weighing of the considerations in the form of various score penalties. A randomized algorithm was presented that attempts to minimize the overall score of the domains to be designed by mutating a few bases at a time, and accepting mutations that lead to improved (lower) scores. The Domain Design (DD) software runs the randomized algorithm for sequence design while allowing the user to dynamically intervene.



```
Domain                         Sequence                              Imp. Comp. Score
    0         1         2         3         4         5         6
    1234567890123456789012345678901234567890123456789012345678901234567890

1   ACACTTACGTTCATT                                                 1    N    17.500010
2   GGTAAACCGTCCCTTTAAGC                                            1    N    17.500020
3   CCCGTAAAAC                                                      1    N    15.000030
4   CCAGTAATCTTC                                                    5    N    19.000040
5   CTTATACAATAC                                                    1    ATC  13.500050
6   CACTACAAATGACTCC                                                1    N    16.000060
7   CTTAACCGAAACACAC                                                1    N    18.000070
8   TTAGAATA                                                        1    GAT  18.000080

Modify (D)omains, modify (B)ases, (S)ave current sequences,                   19.000040
(E)valuate scores, or press space to continue: ▌


Attempts: 13696              Mutations: 1622
```

Fig. 5. Sample run of Domain Design (DD) software. The sequences of the domains are shown in **(A)**, with the bases in red denoting bases that are "locked" and prevented from mutation. The user can lock or unlock bases and domains through the course of the design process, as well as manually changing the sequences of the domains. The numbers in the **(B)** column list the "importance" of each domain, which is implemented as an additive adjustment to the score of the domain. **(C)** shows the nucleotide compositions of each of the domains, with "N" denoting no constraints (all four nucleotides may be used). The overall score of the set of domains is shown in **(D)**. The tiny decimal portion in the overall score shows the number of the domain currently possessing the worst score, to facilitate user intervention. In the figure, Domain 4 current has the worst score.

At its heart, DD is a low-level domain-based approach to sequence design, and does not consider the ways in which the domains are concatenated to form DNA strands, nor the overall architecture of the reaction network using those strands. As a result, the sequences generated by DD will in general not be as good as those generated by software intended specifically for particular applications. The comparative advantage of DD is its speed, simplicity, and generalizability–by ignoring the details of particular strands and systems, it seeks to generate sequences that are "good enough" for a wide variety of DNA nanotechnology applications based on strand displacement. Empirical evidence of the success of DD lie in experimental works published by the author, who used DD to design the sequences for a variety of strand displacement-based reactions and networks exhibiting catalysis [12] [4] [13] [8].

Currently, most experimental work on dynamic DNA nanotechnology have been very limited in scale, with numbers of different DNA strands concurrently in solution being less than 100. At these scales, it is relatively easy to design sequences that do not significantly interact or crosstalk with one another. As a result, many different sequence design software (both published and unpublished) will likely generate sequences that work decently well for their intended applications, and it is not easy to critically evaluate their relative performances. As researchers develop and experimentally test larger reaction networks [40] [52] [53], the demand for large numbers of DNA sequences that must exist stably simultaneously in solution will drive the development of ever better and easy-to-use DNA sequence design software.

Source code for Domain Design can be downloaded at:
http://www.dna.caltech.edu/∼dzhang/softsource/DD.c

Acknowledgements. This research is supported in part by NSF grants 0832824 and 0728703. DYZ is supported by the Fannie and John Hertz Foundation.

The author thanks Erik Winfree and David Soloveichik for testing the DD code. The author thanks Erik Winfree for suggestions regarding the manuscript.

References

1. SantaLucia, J., Hicks, D.: Annu. Rev. Biophys. Biomol. Struct. 33, 415 (2004)
2. Dirks, R.M., Bois, J.S., Schaeffer, J.M., Winfree, E., Pierce, N.A.: SIAM Rev. 49, 65 (2007)
3. Zuker, M.: Nucleic Acids Res. 31, 3406 (2003)
4. Zhang, D.Y., Winfree, E.: J. Am. Chem. Soc. 131, 17303 (2009)
5. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Science 314, 1585 (2006)
6. Hagiya, M., Yaegashi, S., Takahashi, K.: Nanotechnology: Science and Computation, pp. 293–308 (2006)
7. Frezza, B.M., Cockroft, S.L., Ghadiri, M.R.: J. Am. Chem. Soc. 129, 14875 (2007)
8. Zhang, D.Y.: Cooperative DNA strand displacement for DNA quantitation, detection, and logic (submitted, 2010)
9. Xie, Z., Liu, S.J., Bleris, L., Benenson, Y.: Nuc. Acids Res. (2010, doi:10.1093/nar/gkq117)
10. Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Science 318, 1121 (2007)
11. Turberfield, A.J., Mitchell, J.C., Yurke, B., Mills, A.P., Blakey, M.I., Simmel, F.C.: Phys. Rev. Lett. 90, 118102 (2003)
12. Zhang, D.Y., Winfree, E.: J. Am. Chem. Soc. 130, 13921 (2008)
13. Zhang, D.Y., Winfree, E.: Nuc. Acid Res. (2010, pre-published online doi:10.1093/nar/gkq088)
14. Seelig, G., Yurke, B., Winfree, E.: J. Am. Chem. Soc. 128, 12211 (2006)
15. Bois, J.S., Venkataraman, S., Choi, H.M.T., Spakowitz, A.J., Wang, Z.G., Pierce, N.A.: Nuc. Acid Res. 33, 4090 (2005)
16. Green, S.J., Lubrich, D., Turberfield, A.J.: Biophysical Journal 91, 2966 (2006)
17. Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: Nature 406, 605 (2000)
18. Dirks, R.M., Pierce, N.A.: Proc. Nat. Acad. Sci. 101, 15275 (2004)
19. Yin, P., Choi, H.M.T., Calvert, C.R., Pierce, N.A.: Nature 451, 318 (2008)
20. Omabegho, T., Sha, R., Seeman, N.C.: Science 324, 67 (2009)
21. Gao, Y., Wolf, L.K., Georgiadis, R.M.: Nuc. Acids Res. 34, 3370 (2006)

22. Sun, W., Mao, C., Liu, F., Seeman, N.C.: J. Mol. Biol. 282, 59 (1998)
23. Dirks, R.M., Lin, M., Winfree, E., Pierce, N.A.: Nucleic Acids Res. 32, 1392 (2004)
24. Seifferf, J., Huhle, A.: J. Biomol. Struct. Dyn. 25, 453 (2008)
25. Tanaka, F., Kameda, A., Yamamoto, M., Ohuchi, A.: Nuc. Acids Res. 33, 903 (2005)
26. Tulpan, D., Andronescu, M., Chang, S.B., Shortreed, M.R., Condon, A., Hoos, H.H., Smith, L.M.: Nuc. Acids Res. 33, 4951 (2005)
27. Sager, J., Stefanovic, D.: Designing Nucleotide Sequences for Computation: A Survey of Constraints. In: Carbone, A., Pierce, N.A. (eds.) DNA 11. LNCS, vol. 3892, pp. 275–289. Springer, Heidelberg (2006)
28. Seeman, N.C.: J. Biomol. Struct. Dyn. 8, 573–581 (1990)
29. Cao, S., Chen, S.: Nuc. Acids Res. 34, 2634 (2006)
30. Xayaphoummine, A., Bucher, T., Isambert, H.: Nuc. Acids Res. 33, W605 (2005)
31. Dirks, R.M., Pierce, N.A.: J. Comput. Chem. 25, 1295 (2004)
32. Protozanova, E., Yakovchuk, P., Frank-Kamenetskii, M.D.: J. Mol. Biol. 342, 775 (2004)
33. Pyshnyi, D.V., Ivanova, E.M.: Russian Chemical Bulletin 51, 1145 (2002)
34. Vasiliskov, V.A., Prokopenko, D.V., Mirzabekov, A.D.: Nuc. Acid Res. 29, 2303 (2001)
35. Pyshnyi, D.V., Ivanova, E.M.: Nucleosides, Nucleotides, and Nucleic Acids 23, 1057 (2004)
36. Panyutin, I.G., Hsieh, P.: J. Mol. Biol. 230, 413 (1993)
37. Sen, D., Gilbert, W.: Methods in enzymology 211, 191 (1992)
38. Southern, E.M., Casegreen, S.C., Elder, J.K., Johnson, M., Mir, K.U., Wang, L., Williams, J.C.: Nuc. Acids Res. 22, 1368 (1994)
39. Mir, K.U.: A restricted genetic alphabet for DNA computing. In: DNA Based Computers II. DIMACS, vol. 44, pp. 243–246 (1998)
40. Qian, L., Winfree, E.: A Simple DNA Gate Motif for Synthesizing Large-Scale Circuits. In: Goel, A., Simmel, F.C., Sosík, P. (eds.) DNA 14. LNCS, vol. 5347, pp. 70–89. Springer, Heidelberg (2009)
41. Mao, C., Sun, W., Shen, Z., Seeman, N.C.: Nature 397, 144 (1999)
42. Reynaldo, L.P., Vologodskii, A.V., Neri, B.P., Lyamichev, V.I.: J. Mol. Bio. 297, 511 (2000)
43. Yurke, B., Mills, A.P.: Genet. Prog. Evol. Mach. 4, 111 (2003)
44. Zuker, M., Mathews, D.H., Turner, D.H.: Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A Practical Guide. In: Barciszewski, J., Clark, B.F.C. (eds.) RNA Biochemistry and Biotechnology. NATO ASI Series. Kluwer Academic Publishers, Dordrecht (1999)
45. Dimitrov, R.A., Zuker, M.: Biophys. J. 87, 215 (2004)
46. Kim, J., White, K.S., Winfree, E.: Mol. Syst. Biol. 2, 68 (2006)
47. Dittmer, W.U., Simmel, F.C.: Nano Lett. 4, 689 (2004)
48. Stojanovic, M.N., Semova, S., Kolpashchikov, D., Macdonald, J., Morgan, C., Stefanovic, D.: J. Am. Chem. Soc. 127, 6914–6915 (2005)
49. Pei, R., Taylor, S.K., Stefanovic, D., Rudchenko, S., Mitchell, T.E., Stojanovic, M.N.: J. Am. Chem. Soc. 128, 12693 (2006)
50. Lund, K., Manzo, A., Dabby, N., Michelotti, N., Johnson-Buck, A., Nangreave, J., Taylor, S., Pei, R., Stojanovic, M.N., Walter, N., Winfree, E., Yan, H.: Nature (in press, 2010)
51. Frank-Kamenetskii, M.D., Mirkin, S.M.: Annu. Rev. Biochem. 64, 65 (1995)
52. Soloveichik, D., Seelig, G., Winfree, E.: Proc. Nat. Acad. Sci. (2010, pre-published online doi:10.1073/pnas.0909380107)
53. Phillips, A., Cardelli, L.: Journal of the Royal Society Interface 6, S419 (2009)

DNA-Based Fixed Gain Amplifiers and Linear Classifier Circuits

David Yu Zhang[1] and Georg Seelig[2]

[1] California Institute of Technology, Pasadena, CA, USA
[2] University of Washington, Seattle, WA, USA
dzhang@dna.caltech.edu, gseelig@u.washington.edu

Abstract. DNA catalysts have been developed as methods of amplifying single-stranded nucleic acid signals. The maximum turnover (gain) of these systems, however, often varies based on strand and complex purities, and has so far not been well-controlled. Here we introduce methods for controlling the asymptotic turnover of strand displacement-based DNA catalysts and show how these could be used to construct linear classifier systems.

1 Introduction

DNA nanotechnology has utilized the specific binding properties [1] and the well-understood thermodynamics [2] and kinetics [3] [4] of nucleic acids to construct dynamic cascaded reactions, such as logic gates and circuits [5] [6] [7] [8], motors [9] [10], and amplification mechanisms [11] [12] [13] [14] [15] [16] [17].

DNA devices can operate in complex biochemical environments and can be programmed to specifically interact with biological nucleic acids such as messenger RNA (mRNA) or microRNA (miRNA). DNA circuits could be used to develop novel point-of-care diagnostic devices that integrate detection with analysis and do not require complex laboratory equipment. It has even been suggested to use DNA devices as "smart therapeutics" that operate inside living cells and integrate detection of specific disease markers with the activation of a therapeutic response based on the RNA interference pathway [18] [19], on antisense oligonucleotides [20] or ribozymes.

Such applications require nucleic acid circuitry that can reliably identify a specific disease state. Characteristic RNA markers that could serve as inputs to a DNA analytic circuit have been identified for many diseases. However, it is often not sufficient to simply detect the presence or absence of a set of RNA markers. Instead the classifiers that distinguish a disease tissue from healthy tissue (or other disease tissues) are often complex functions of the concentrations of multiple RNA markers (see Refs. [21] [22] for examples of microRNA-expression based classifiers of varying complexity).

Here we propose a molecular implementation for a specific class of classifiers, namely linear classifiers. The classifier circuit computes a linear combination

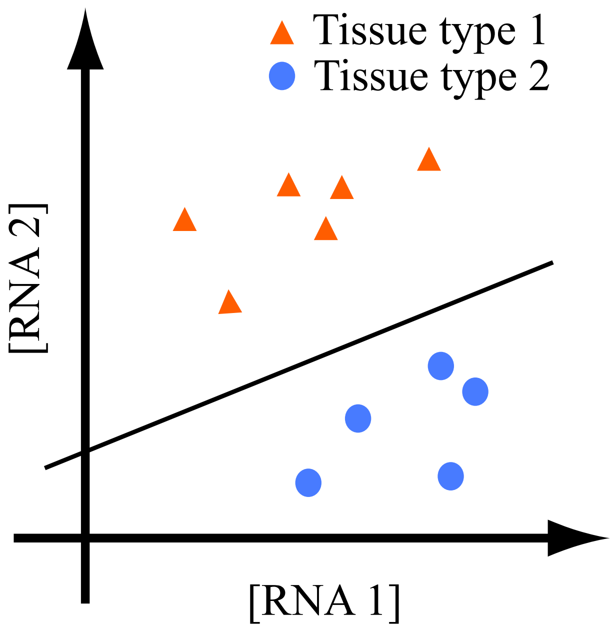


Fig. 1. Sketch of a hypothetical two-gene classifier. Samples from two different tissue can be clearly distinguished based on the expression profiles of two RNA molecules.

with arbitrary (positive or negative) weights on a set of inputs (e.g. RNA molecules) and compares the result to a threshold value. Fig. 1 shows a highly simplified sketch of a linear two-gene classifier: the line separating the two different tissue types is given by an equation of the form $\alpha_1[RNA_1]+\alpha_2[RNA_2]=K$. Given a sample of unknown origin, we can now classify it as tissue type 1 or 2 based on a measurement of two RNAs. Unlike in the more conventional case where the expression of each RNA is individually measured and the linear classification analysis is performed *in silico*, here both detection and analysis are done on the molecular level, allowing *in situ* and *in vivo* applications.

Previous DNA logic gates and circuits were mostly designed for a situation where inputs can be represented as Boolean variables and are either present at a high concentration or completely absent [8] [5]. This does not necessarily require the original inputs to be at a specific level; DNA-based signal restoration units consisting of a threshold gate and an amplifier can be used to restore an input with an arbitrary concentration to the expected logical TRUE or FALSE values. Still, the digital nature of such circuits is inherently incompatible with classification problems, in which the relative amounts of inputs determines the value of the final output. The fixed gain amplification methods presented here allow reliable tuning of analog signals encoded in the concentrations of nucleic acids.

2 Fixed Gain Amplifiers: Lowering Catalytic Turnover

One key component of the proposed linear classifier is a DNA-based catalytic amplifier, that allows one signal-stranded nucleic acid to specifically produce or release many single-stranded nucleic acid molecules of independent sequence. Importantly, this amplifier needs to have a finite and controllable gain α such that each input on average releases α copies of the output. Such a finite gain amplifier would be useful not only in a linear classifier, where each detected RNA species is assigned a different weight, but could also be used for pre-amplification of a set of low-concentration inputs while maintaining their relative concentrations.

Existing DNA amplifiers have an intrinsically finite turnover; strand displacement-based nucleic acid catalysts typically convert on the order of 10-100 substrates before being inactivated [14] [15]. Inactivation is most likely due to defective substrate complexes or fuels [14] [15] [17]. The details of the inactivation process depend on the specifics of the amplifier design, but it seems likely that imperfectly synthesized DNA strands are a major culprit. In practice, the maximal turnover obtained seems to depend strongly on sequence, purification procedures, strand orientation and similar experimental and design details. Therefore, while the gain is finite, it can be characterized for any particular system.

The question then becomes if, starting from an arbitrary but high turnover, we can lower the turnover controllably to a fixed value. Given the intrinsic turnover of a catalytic system, intuitively it seems clear that we can lower the turnover further either by increasing the fraction of imperfect substrate or through addition of an alternative competitive inhibitor that irreversibly binds to the catalyst. However, it may be less intuitive how to best adjust the turnover to any specific desired value.

To address the question of how to control turnover we first consider a simple model for a catalytic reaction with competitive inhibition. Afterwards, we simulate a specific DNA implementation using measured reaction parameters. A catalytic reaction in the presence of an impurity can be modeled as:

$$C + S \xrightarrow{k_a} C + P \tag{1}$$

$$C + D \xrightarrow{k_b} \emptyset \tag{2}$$

In the first reaction a catalyst C transforms a substrate S into a product P. The rate constant for this reaction is k_a. The catalyst can also participate in a second, unproductive reaction with an inhibitor (or damper) D. This reaction proceeds at a rate constant k_b.

The differential equations resulting from this model can be integrated with initial conditions $C(0) = C_0$, $S(0) = S_0$, $D(0) = D_0$ and $P(0) = 0$. Solving for the product $P(t)$ we get

$$P(t) = S_0 - S_0 \left(\frac{1-\rho}{1-\rho e^{k_b \Delta t}} \right)^{k_a/k_b} \tag{3}$$

where we introduced the ratio $\rho = C_0/D_0$ and the difference $\Delta = C_0 - D_0$ of the initial amounts of catalyst and inhibitor.

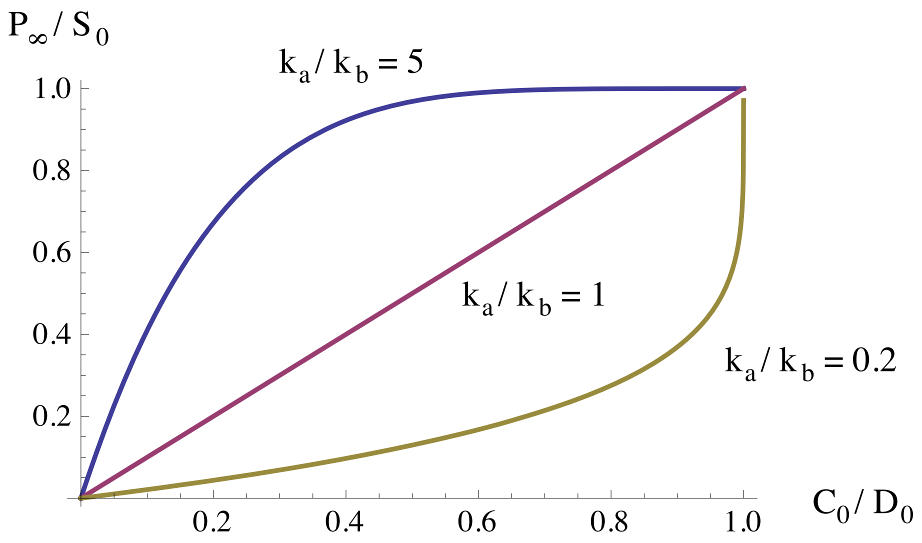


Fig. 2. Output produced in a catalytic reaction with competitive inhibition as a function of the catalyst concentration. Final product P_∞ is scaled by initial concentration of substrate S_0. Initial catalyst concentration C_0 is measured in units of inhibitor concentration I_0. We obtain different I/O characteristics depending on the ratio between the rate constants k_a and k_b for the catalytic and the competitive reaction.

In an ideal system without competitive inhibition the final product concentration is always equal to the initial concentration of substrate. Given enough time the catalyst will convert all substrate into product. In a system with competitive inhibition this is not necessarily true. The final amount of product produced in that case can be computed by taking the limit $t \to \infty$ in Eq. 3:

$$\lim_{t\to\infty} P(t) = P_\infty = \begin{cases} S_0, & C_0 \geq D_0 \\ S_0 - S_0\,(1-\rho)^{k_a/k_b}, & C_0 < D_0 \end{cases} \tag{4}$$

Not surprisingly, if we start out with more catalyst than inhibitor, the reaction will eventually go to completion. The opposite limit is more interesting.

First, consider the case where the rate for the catalytic reaction is much faster than the inhibition reaction, $k_a > k_b$ (blue trace in Fig. 2). In this case the inhibitor has a relatively minor effect that is most pronounced at low concentrations of catalyst compared to the inhibitor.

In the limit where the catalytic reaction occurs at exactly the same rate as the inhibitory reaction, i.e. $k_a = k_b$ (red trace in Fig. 2) Eq. 4 predicts that the final amount of product is linear in the initial amount of catalyst, i.e. $P_\infty = \alpha C_0$ where $\alpha = S_0/D_0$. That is, by adjusting the relative concentration of substrate to inhibitor we can get any finite gain we need.

$S + C \underset{k_2}{\overset{k_1}{\rightleftharpoons}} I1 + SP$	$k_0 = 5\ \mathrm{M}^{-1}\ \mathrm{s}^{-1}$
$I1 + F \xrightarrow{k_2} I2 + OP$	$k_1 = 2.7 \cdot 10^5\ \mathrm{M}^{-1}\ \mathrm{s}^{-1}$
$I2 \underset{k_1}{\overset{k_3}{\rightleftharpoons}} W + C$	$k_2 = 1.1 \cdot 10^6\ \mathrm{M}^{-1}\ \mathrm{s}^{-1}$
$S + F \xrightarrow{k_0} OP + SP + W$	$k_3 = 1.1 \cdot 10^{-2}\ \mathrm{s}^{-1}$
$I1 + Fb \xrightarrow{k_2} X + OP$	$k_{rep} = 4 \cdot 10^5\ \mathrm{M}^{-1}\ \mathrm{s}^{-1}$
$S + Fb \xrightarrow{k_0} OP + SP + W2$	
$C + W2 \xrightarrow{k_1} X$	
$C + D \xrightarrow{k_1} X$	
$A + C \underset{k_2}{\overset{k_1}{\rightleftharpoons}} IA1 + SP$	
$IA1 + F \xrightarrow{k_2} I2 + C$	
$I2 \underset{k_1}{\overset{k_3}{\rightleftharpoons}} W + C$	
$A + F \xrightarrow{k_0} C + SP + W$	
$IA1 + Fb \xrightarrow{k_2} X + C$	
$A + Fb \xrightarrow{k_0} C + SP + W2$	

Table 9-1: Reactions simulated in Fig. 9-4.

The situation where the rate for the inhibitor reaction is faster than the rate for the catalytic reaction is also interesting. In that case, the amount or product is sub-linear in the initial amount of catalyst for $C_0 < D_0$ but reaches a fixed value S_0 in the opposite regime. The concentration of the competitive inhibitor I therefore acts as a threshold for the catalytic reaction. Such a threshold element is useful for reliable signal propagation for example in the context of chemical digital circuits.

We now turn to a specific DNA implementation of such a system. Our implementation is based on the entropy-driven catalytic amplifier of Ref. [15] which was further characterized in Ref. [17]. Turnover for this amplifier was measured to be about 100. The reaction mechanism for this system including the side reactions leading to intrinsically finite turnover is shown in Fig. 3A. A reaction between catalyst strand and substrate relies on toehold mediated strand displacement. As a competitive inhibitor we here propose to use a damper DNA gate that irreversibly binds the catalytic input (Fig. 3B). In order to match the reaction rate constants of the catalyst with this inhibitor to that of the catalyst with the active substrate we simply choose the toeholds for both reactions to be identical.

In order to verify the predictions from our simple model Eq. 1 we simulated the full catalytic system of Ref. [15] with a parallel inhibitory reaction using the measured rate constants and reaction intermediates. The model is given in Table 9-1 and resulting data is shown in Fig 4A. As expected from our model, the final fluorescence depends linearly on the concentration of the damper gate.

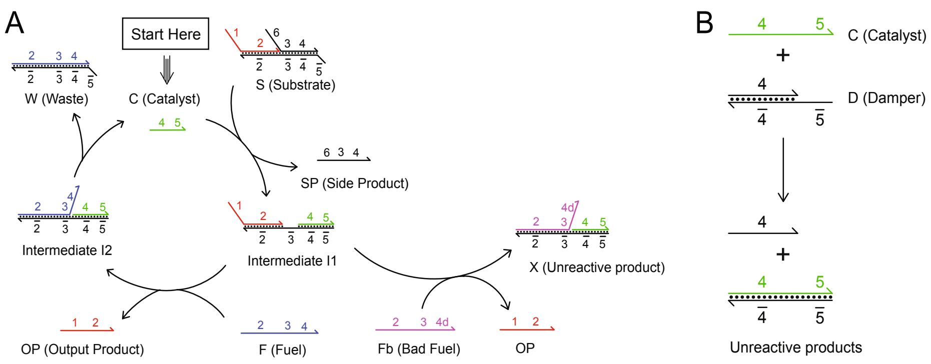


Fig. 3. Methods for tuning catalytic turnover. **(A)** DNA amplification via catalysis, adapted from Zhang et al. [17]. Catalyst strand C reacts with S to form side product SP and intermediate $I1$, the latter of which subsequently reacts with F to release output product OP, waste W, and catalyst C. However, a small fraction of bad fuel with deletions and/or degradation near the 3' end, denoted as Fb, will bind to intermediate $I1$ to form an unreactive product X, thus permanently trapping catalyst C and reducing the observed catalytic turnover of the reaction. The ratio $\frac{[Fb]}{[F]+[Fb]}$ was estimated to be 0.01 for HPLC-purified fuel strands [17]. **(B)** The catalytic turnover of the reaction can be tuned to be lower via the addition of the damping complexes D. Because C binds by the toehold to D as to S, it is assumed that this rate constant is identical in value to that of k_1. **(C)** Schematic of a generalized catalytic reaction with arbitrary control over turnover. In the original work on entropy-driven DNA catalysts [15], it was shown that the catalytic reaction can be made autocatalytic by using an alternative substrate A, which releases as product an molecule identical to the catalyst. Turnover can be increased above the limit set by fuel purity with autocatalytic substrate.

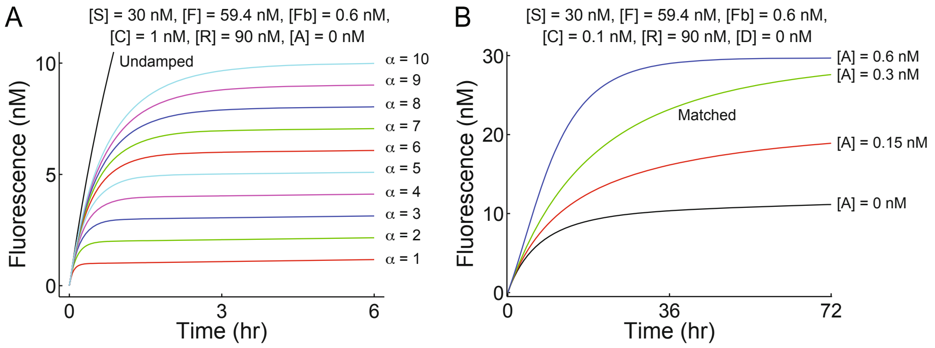


Fig. 4. Modulating turnover. **(A)** Simulations of the entropy-driven catalyst system with damper. [17]. Various amounts of D were present to achieve the fixed turnover η shown, with $[D] = \frac{30}{\alpha} - 0.3$ nM. See Table 9-1 for the full set of simulated reactions. **(B)** Simulations of the entropy-driven catalyst system with autocatalytic substrate. At $[A]$ = 0.3 nM, the increase of catalyst due the autocatalytic substrate nearly matches the decrease of the catalyst due to bad fuel. With lower concentrations of A, asymptotic turnover is limited. With higher concentrations of A, the reaction adopts autocatalytic characteristics, and becomes less sensitive to the initial concentration of the catalyst.

3 Fixed Gain Amplifiers: Increasing Catalytic Turnover

The turnover of a catalytic reaction can be increased above the intrinsic limit set by defective oligonucleotides. It seems clear that it should be possible to compensate for the loss of catalyst in an unproductive reaction through the production of an extra catalyst in a parallel autocatalytic reaction that proceeds at the same rate. A simple model motivated by this intuition is

$$C + S \xrightarrow{k_a} C + P, \tag{5}$$

$$C + D \xrightarrow{k_b} \emptyset, \tag{6}$$

$$C + A \xrightarrow{k_b} 2C. \tag{7}$$

Here A is the substrate for the autocatalytic reaction which is present initially at a concentration $A(0) = A_0$. With the same initial conditions as above we can solve the resulting differential equation. The final product as a function of time then is

$$P(t) = S_0 - S_0 \left(\frac{1 - \sigma}{1 - \sigma e^{k_a \Gamma t}} \right)^{k_a/k_b}, \tag{8}$$

where $\Gamma = C_0 + A_0 - D_0$ and $\sigma = C_0/(D_0 - A_0)$. The result is therefore of exactly the same form as Eq. 3 if we make the substitution $D_0 \rightarrow D_0 - A_0$. In the special case where the initial concentrations of the inhibitor D and the substrate A for the autocatalytic reactions are the same, i.e. $A_0 = D_0$, these reactions cancel each other out and $P(t) = S_0 \exp(-k_a C_0 t)$ as expected for an ideal catalytic reaction. If $A_0 > D_0$ the overall kinetics of the reaction is that of an autocatalytic reaction. In fact, for $k_a = k_b$ Eq. 8 looks very similar to the logistic equation we obtain when solving a simple autocatalytic reaction. The different limiting cases for the amount of product P_∞ for $t \rightarrow \infty$ follow from the discussion above if we make the substitution $D_0 \rightarrow D_0 - A_0$.

4 A Linear Classifier

Based on the fixed gain amplifier systems explained above we can now build a linear classifier that implements a function

$$\sum_i \alpha_i [m_i] = T. \tag{9}$$

Here α_i are the weights, $[m_i]$ the concentrations of the molecular species m_i and K is the threshold. A molecular implementation of this function thus requires that an initial concentration of m_i results in a concentration $\alpha_i [m_i]$ of some signal molecules that can be compared to each other and to the concentration K of a threshold molecule.

An element of the sum with a positive weight α_i is implemented as a catalytic reaction with a fixed gain α_i. An input m_i at initial concentration $[m_i]_0$ results

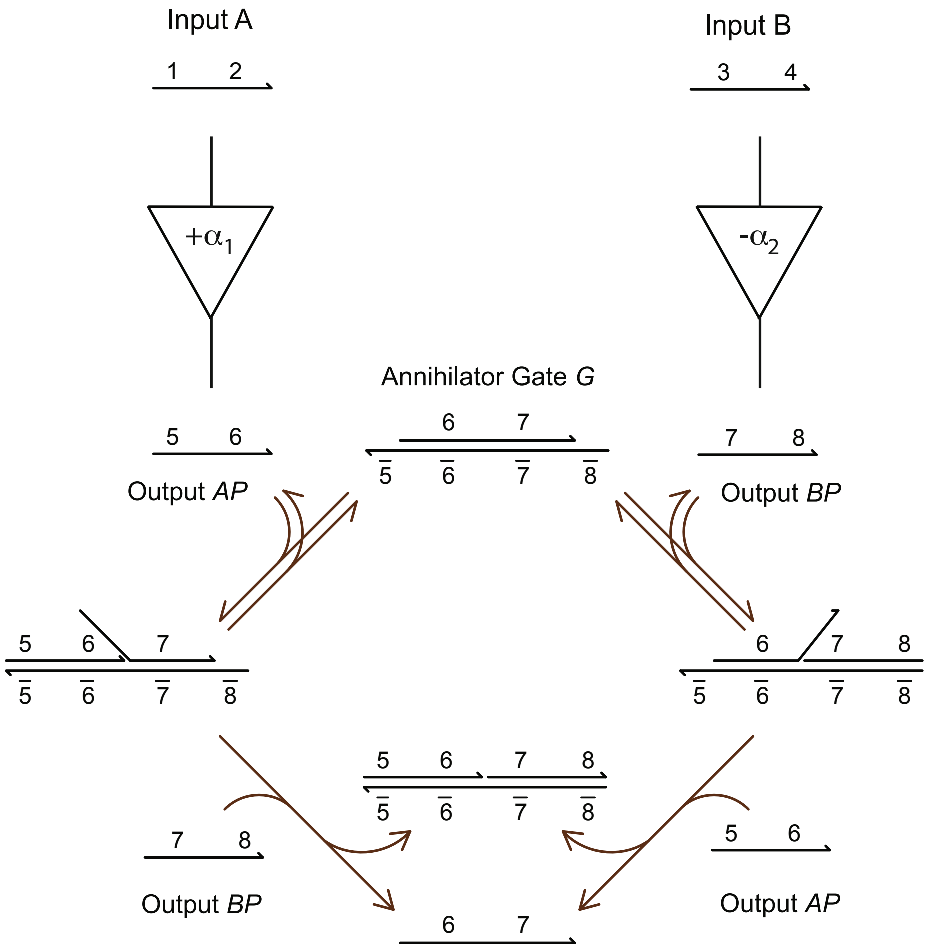


Fig. 5. Implementing negative gain. We implement negative gain by having all inputs with positive gains catalytically produce one product AP, and all inputs with negative gains catalytically produce another product of independent sequence, BP. The products AP and BP stoichiometrically neutralize one another via the annihilator gate AG [24]. Excess AP at the end of the reaction denotes that the density classification expression evaluated to positive, while excess BP denotes the expression evaluated to negative.

in a final concentration $\alpha_i[m_i]_0$ of an output strand AP of unrelated sequence. Importantly, the output strand is the same for all reactions with a positive α_i. Similarly, every reaction with a negative α_i is implemented as a catalytic reaction with a (positive) gain $|\alpha_i|$ but a different output strand BP.

In principle, we could use reporters with two different colors to independently read out the the positive and negative output strands AP and BP. Using fluorescence calibration curves, we could then compute the respective concentrations

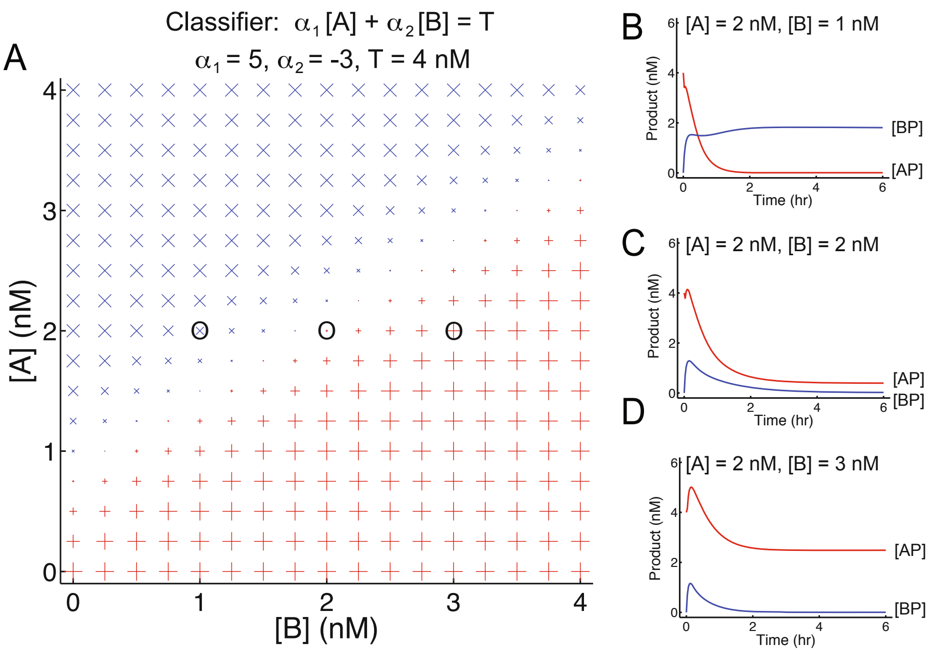


Fig. 6. DNA classifier. **(A)** Summary plot of the concentrations of AP and BP at the end of 6 hours of simulated reaction for various initial concentrations of A and B. Size of crosses denote the final concentration of AP; size of pluses denote final concentration of BP. **(B), (C), (D)** Sample concentration traces for AP and BP.

as well as the difference between them and compare the result to the threshold value T. However, such an approach would still require considerable intervention form an experimentalist meaning that only part of the computation is actually implemented as molecular computation.

To embed the comparison of the concentrations of AP and BP in the DNA molecules themselves, we use the annihilator gate design presented in Ref [24] (see also Fig. 5). In this design, each of AP and BP bind to annihilator gate G reversibly, but the combination of the two irreversibly binds to G, removing both from solution (Fig. 9-5). In an excess of annihilator gate G, only one of AP and BP will be present in solution at significant concentration. G is present in solution from the beginning of the beginning of the reaction, and serves to dynamically reduce the concentrations of both AP and BP. Note that a similar mutual annihilation reaction could also be implemented using the mechanism for implementing arbitrary bimolecular reactions explained in Ref. [23].

So far we have shown how to implement arbitrary positive and negative gains and how to perform molecular-level comparison of the concentrations of the resulting reporter strands AP or BP. This would be sufficient to implement a classifier with $T = 0$. To implement a non-zero value for the threshold T we simply add T units of AP or BP depending on the sign of T. In this way we can implement a molecular classifier with arbitrary values for α_i and T on the molecular level.

Fig. 6 shows an example of a simulation of a simple two-input linear classifier. The simulations use a realistic model for the underlying DNA reactions. Fig. 6A shows the expected final signal (i.e. the excess amount of AP or BP) for a variety of "samples." Each sample is characterized by a pair (A, B) of the two molecules of interest. Note that without further amplification of the final output (either AP or BP) the signal linearly increases with the distance from the threshold line.

5 Conclusions

Here we have proposed a DNA implementation of a fixed gain amplifier and of linear classifier circuits. The fixed gain amplifier combines a DNA catalytic amplifier with a threshold element or an autocatalytic reaction in order to obtain arbitrary gain that can be lower or higher than the intrinsic gain of the DNA catalyst. Classifier circuits similar to the one proposed here can potentially be used for the embedded analysis of RNA expression levels in complex mixtures. Such classification circuits could find applications in point-of-care diagnostics or could even be used to analyze gene expression in living cells.

To apply the presented linear classifier circuit to actual cell state classification, however, the classifier must be able to deal with RNA input concentrations that are often low and can vary by orders of magnitude. While in theory the methods presented should be able to allow indefinitely high values of α, the precise control of large values of α will be difficult in practice, because the intrinsic turnover set by strand purities will not be known to great accuracy. Additionally, achieving high turnover will be slow, because each turnover requires a fixed amount of time for reaction.

Multi-stage fixed turnover amplifiers can be used to combat the aforementioned difficulties. That is, the products AP and BP can be themselves amplified by another fixed gain amplifier, and the gains of the two systems will be multiplied. Achieving high turnovers with a 2-stage system will also be quadratically faster. For extremely high turnovers, even more stages of fixed amplification can be cascaded.

There are a variety of alternatives to the specific implementation proposed here. In particular, the chemical reaction systems networks of Ref. [23] can be used to implement the reactions described here. However, the catalytic system of Ref. [15] is currently the best characterized and also fastest catalytic amplifier available which is why we chose to use this system for our design.

The reactions and mechanisms used to construct the linear classifier have either been demonstrated or are similar enough to well-understood reactions that they are expected to experimentally function as designed. All simulation results shown include modeling of relevant intermediate species and side reactions; similar modeling has been able to quantitatively predict the kinetics of similar DNA constructions [4] [17]. Thus, we are optimistic that we can experimentally demonstrate the density classifier circuit *in vitro* in the near future.

Acknowledgments. DYZ is supported by the Fannie and John Hertz Foundation. GS is supported by a Career Award at the Scientific Interface from the Burroughs Wellcome Fund and an NSF CAREER award.

References

1. Bloomfield, V.A., Crothers, D.M., Tinoco Jr., I.: Nucleic Acids: Structures, Properties, and Functions. University Science Books, Sausalito (2000)
2. SantaLucia, J., Hicks, D.: Annu. Rev. Biophys. Biomol. Struct. 33, 415 (2004)
3. Yurke, B., Mills, A.P.: Genet. Prog. Evol. Mach. 4, 111 (2003)
4. Zhang, D.Y., Winfree, E.: J. Am. Chem. Soc. 131, 17303 (2009)
5. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Science 314, 1585 (2006)
6. Hagiya, M., Yaegashi, S., Takahashi, K.: Nanotechnology: Science and Computation, pp. 293–308 (2006)
7. Frezza, B.M., Cockroft, S.L., Ghadiri, M.R.: J. Am. Chem. Soc. 129, 14875 (2007)
8. Qian, L., Winfree, E.: A Simple DNA Gate Motif for Synthesizing Large-Scale Circuits. In: Goel, A., Simmel, F.C., Sosík, P. (eds.) DNA 14. LNCS, vol. 5347, pp. 70–89. Springer, Heidelberg (2009)
9. Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: Nature 406, 605 (2000)
10. Bath, J., Turberfield, A.J.: Nat. Nanotech. 2, 275 (2007)
11. Turberfield, A.J., Mitchell, J.C., Yurke, B., Mills, A.P., Blakey, M.I., Simmel, F.C.: Phys. Rev. Lett. 90, 118102 (2003)
12. Bois, J.S., Venkataraman, S., Choi, H.M.T., Spakowitz, A.J., Wang, Z.G., Pierce, N.A.: Nuc. Acid Res. 33, 4090 (2005)
13. Green, S.J., Lubrich, D., Turberfield, A.J.: Biophysical Journal 91, 2966 (2006)
14. Seelig, G., Yurke, B., Winfree, E.: J. Am. Chem. Soc. 128, 12211 (2006)
15. Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Science 318, 1121 (2007)
16. Yin, P., Choi, H.M.T., Calvert, C.R., Pierce, N.A.: Nature 451, 318 (2008)
17. Zhang, D.Y., Winfree, E.: Nuc. Acid Res. (2010, pre-published online doi:10.1093/nar/gkq088)
18. Masu, H., Narita, A., Tokunaga, T., Ohashi, M., Aoyana, Y., Sando, S.: Angew. Chemie Int. Ed. 48, 9481 (2009)
19. Xie, Z., Liu, S.J., Bleris, L., Benenson, Y.: Nuc. Acids Res. (2010, doi:10.1093/nar/gkq117)
20. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: Nature 429, 423 (2004)
21. Lu, J., et al.: Nature 435, 834 (2005)
22. Rosenfeld, N., et al.: Nat. Biotech. 26, 462 (2008)
23. Soloveichik, D., Seelig, G., Winfree, E.: Proc. Nat. Acad. Sci. (2010, pre-published online doi:10.1073/pnas.0909380107)
24. Zhang, D.Y.: Cooperative DNA strand displacement for DNA quantitation, detection, and logic (submitted, 2010)

Author Index