Universal Computation via Self-assembly of DNA: Some Theory and Experiments

Erik Winfree[1]*, Xiaoping Yang[2], Nadrian C. Seeman[2]

August 7, 1996

[1] Computation and Neural Systems Option, California Institute of Technology
[2] Department of Chemistry, New York University

Abstract

In this paper we examine the computational capabilities inherent in the hybridization of DNA molecules. First we consider theoretical models, and show that the self-assembly of oligonucleotides into linear duplex DNA can only generate sets of sequences equivalent to regular languages. If branched DNA is used for self-assembly of dendrimer structures, only sets of sequences equivalent to context-free languages can be achieved. In contrast, the self-assembly of double crossover molecules into two dimensional sheets or three dimensional solids is theoretically capable of universal computation. The proof relies on a very direct simulation of a universal class of cellular automata. In the second part of this paper, we present results from preliminary experiments which investigate the critical computational step in a two-dimensional self-assembly process.

1 Introduction

A fundamental property of DNA is that, under the right conditions, Watson-Crick complementary regions of single-stranded DNA will hybridize and form a double helical structure. This property, in vitro and in vivo, can lead DNA to assume a remarkable diversity of geometric forms[1]. Under certain simplifying conditions, the behavior of hybridization is sufficiently predictable to be considered as a computational primitive; i.e., a function from initial oligonucleotides to final supramolecular structures is computed. The computational aspects of self-assembly were exploited for the first time in [Adleman], where linear self-assembly was used as a step in solving the Hamiltonian Path Problem. When the self-assembly of tree-like structures takes place, due to the presence of branched junctions, a slightly more powerful computation results. We review a two dimensional generalization capable of universal computation, as suggested in [Winfree], and also suggest a concrete three dimensional self-assembly process.

*To whom correspondence should be addressed (winfree@hope.caltech.edu). Erik Winfree has been supported in part by National Institute for Mental Health (NIMH) Training Grant # 5 T32 MH 19138-06; also by General Motors' Technology Research Partnerships program and by the Center for Neuromorphic Systems Engineering as a part of the National Science Foundation Engineering Research Center Program under grant EEC-9402726. The experimental portion of this research has been partially supported by grants N00014-89-J-3078 from the Office of Naval Research and GM-29554 from the NIH (to NCS).

[1] In vivo, not only is there single-stranded and double-stranded DNA, but branched junctions are formed during recombination, and trypanosomes maintain complex networks of circular DNA within which RNA editing occurs.

2 Computational Power of Abstracted Hybridization

In order to understand the computational implications of DNA hybridization, we will first consider a highly abstracted mathematical model. The physical system we would like to model can be described as follows:

> Synthesize several sequences of DNA. Mix the DNA together in solution. Heat it up and slowly cool it down, allowing complexes of DNA to form. Chemically or enzymatically ligate adjacent strands. Denature the DNA again, and ask, what single-stranded DNA sequences are now present in the solution?

A proper answer to this question is beyond our capability, and realistically detailed models might not be enlightening regarding the logical essence of self-assembly. We therefore investigate very simple models, which, nonetheless, are sufficiently realistic that translation into real world scenarios should be direct. We will consider a number of properties which DNA self-assembly may be postulated to obey, and we will analyze the computational capability and the limits of any self-assembly process which obeys those properties.

Informally, the properties we consider are:

1. **Constant Temperature.** The number of base-pairs required for the stability of DNA complexes does not change during the course of the self-assembly. We thus don't consider annealing, where at high temperatures only long regions will hybridize but later at lower temperatures even short regions can hybridize, but rather we model a "constant temperature" process.
2. **Perfect Watson-Crick Complementarity.** Hybridization only occurs between sequences with perfect Watson-Crick complementarity. Hybridization of mismatching sequences, or that which creates bubbles, branched junctions, triple helices, and other unusual structures, is not considered.
3. **Permanant Binary Events.** All self-assembly interactions occur between two complexes at a time, and no more. These interactions are exclusively hybridizations, joining two complexes together. Furthermore, in the model once two complexes join, they never dissociate.
4. **No Intramolecular Events.** A DNA complex which has self-assembled will not interact with itself, for example by cyclizing. Note, however, that some physically intramolecular interactions can be modeled as a part of a binary event, as discussed below.
5. **Single vs Multiple Binding Regions per Event.** We will consider two cases: either (a) each binary hybridization event creates a single contiguous Watson-Crick region, else (b) the binary events may result in the formation of several physically separated hybridized regions between the two complexes. The latter case is meant to model physical situations where an intermolecular hybridization is immediately followed by an intramolecular hybridization. The case we are interested in is discussed in Section 2.5 (see Figure 7).

6. **Specified Classes of Initial Complexes.** Because of our constant-temperature assumption, it becomes useful to assume that some complexes have already formed prior to the stage of self-assembly which we will consider. Later in the paper, we will consider initial complexes which consist of (a) oligonucleotides, (b) duplex DNA with sticky ends, (c) hairpins with sticky ends, (d) three-armed junctions with sticky ends, (e) double crossover molecules with hairpins and sticky ends, and (f) arbitrary complexes.

Properties (1), (2) and (3) are used primarily for logical simplicity. If Property (4) were changed to allow intramolecular events, it is possible that some of our results would be slightly modified. We will analyze how our results change under different choices for Properties (5) and (6). In Section 2.5, we impose an additional property in order to incorporate geometrical considerations for lattice self-assembly.

2.1 Language Theory and Grammars

Before we present our model of DNA self-assembly, we should comment on what it means to compute by self-assembly. As mentioned above, the typical case is that one starts with a small variety of synthesized oligonucleotides, and one ends with great variety of self-assembled strands. The resulting strands are not random; they have certain properties that derive from being formed from the original oligonucleotides according to certain rules of hybridization.

An analogous situation arises in formal language theory, which has been well understood for many years. There, rather than test tubes of strands, one is interested in sets of symbolic strings, and in methods of generating them. We will sketch the basics here; for a full development see [Ginsburg].

An *alphabet* is a finite set of symbols, for example $\{A, C, G, T\}$ or $\{0, 1\}$ or $\{x, y, z, (,), +, *\}$. A *string* over an alphabet is a finite sequence of symbols from the given alphabet, for example $TATAA$ or 101011 or $(x+y)*z$. A *language* is a well-defined, possibly infinite set of strings, for example $\{$*all strings over* $\{C, T\}$ *of length* 70$\}$ or $\{$*all prime numbers, written in binary*$\}$ or $\{$*all well-formed formulas over* $\{x, y, z, (,), +, -\}\}$.

Although one cannot write down each and every string in an infinite language, one can ask the membership question: is string x in language $\mathcal{L}$? Note that if the language $\mathcal{L}$ contains all bit strings x for which function $f(x) = 1$, the the membership question is equivalent to boolean function evaluation. The membership question may be harder or easier to answer, depending on x and $\mathcal{L}$. Formal language theory goes to great pains to classify languages according to how fancy the computer must be to answer the membership problem. We sketch the fundamental result due to Noam Chomsky, known as the language hierarchy. This requires formalizing the specification of languages by generative rules.

A *rewriting rule* $x \to y$, where x and y are strings, specifies that a string $s = axb$ can be rewritten to produce the new string $s' = ayb$. A *grammar* G is a collection of rewriting rules together with a division of the alphabet into two groups: *terminal symbols* and *nonterminal symbols*, where only nonterminals appear on the left hand side of rewriting rules. Each grammar uniquely defines a language $\mathcal{L}_G$ as follows: the string of terminals s is in $\mathcal{L}_G$ iff it can be obtained from the special nonterminal S by the repeated application of rewriting rules in some order (called a *derivation*).

Grammars may be classified by what kinds of rules they use. We give examples of the three main classes below:

Regular grammars use rules of the form $A \to pB$ and $A \to p$ where A and B are nonterminal symbols and p is a string of terminals. Languages generated by regular grammars are called regular languages. For example, consider the regular grammar $G_E = \{S \to 0S, S \to 1T, S \to 0, T \to 0T, T \to 1S, T \to 1\}$ where 0 and 1 are terminals. This grammar gives rise to all bit strings with an even number of 1's. $101011 \in \mathcal{L}_{G_E}$ because $S \to 1T \to 10T \to 101S \to 1010S \to 10101T \to 101011$. Note that during the derivation we always have a single nonterminal at the right, where all the action takes place. Despite their apparent simplicity, regular languages have found extensive use in pure and applied computer science, perhaps because their membership question can always be answered by an exceedingly simple abstract computer known as a finite state machine.

Context-free grammars use rules of the form $A \to P$ where again A is a nonterminal symbol, but now P is an arbitrary string of terminals and nonterminals. Languages generated by context-free grammars are called context-free languages. Consider the grammar $G_F = \{S \to S+S, S \to M, M \to M*M, M \to (S), M \to x, M \to y, M \to z\}$ where the terminals are $\{x, y, z, (,), +, *\}$. This grammar gives rise to well-formed formulas. $(x + y) * z \in \mathcal{L}_{G_F}$ because $S \to M \to M*M \to M*z \to (S)*z \to (S+S)*z \to (S+M)*z \to (S+y)*z \to (M+y)*z \to (x+y)*z$. Note that whereas it is impossible to generate regular languages whose strings all have long-range structure, one can generate long-range "nested" structure in a context-free language – for example, every parenthesis must be matched in the formulas above. Context-free languages include regular languages. The membership question for context-free languages can be answered by a slightly more complex machine known as a nondeterministic pushdown automaton.

Unrestricted grammars use rules of the form $A \to P$ where now A may be an arbitrary strings of nonterminals, and P is an arbitrary string of terminals and nonterminals. Languages generated by unrestricted grammars are called *recursively enumerable* languages because they include every language which can be generated (enumerated) by any computational process (recursion). Recursively enumerable languages include context-free languages, regular languages, and much more. They are as fancy as you can get. A very simple example: consider the alphabet $\{S, L, R, \overleftarrow{B}, \overrightarrow{B}, \overleftarrow{W}, \overrightarrow{W}, 0, 1\}$ and the grammar $G_P = \{S \to 1, S \to LR, L \to L\,\overrightarrow{B}, L \to 1, R \to \overleftarrow{B}\,R, R \to 1, \overrightarrow{B}\overleftarrow{B} \to \overleftarrow{W}\overrightarrow{W}, \overrightarrow{B}\overleftarrow{B} \to 0, \overrightarrow{B}\overleftarrow{W} \to \overleftarrow{B}\overrightarrow{B}, \overrightarrow{B}\overleftarrow{W} \to 1, \overrightarrow{W}\overleftarrow{B} \to \overleftarrow{B}\overrightarrow{B}, \overrightarrow{W}\overleftarrow{B} \to 1, \overrightarrow{W}\overleftarrow{W} \to \overleftarrow{W}\overrightarrow{W}, \overrightarrow{W}\overleftarrow{W} \to 0\}$ where the terminals are 0 and 1. This gives rise to the rows of Pascal's triangle mod 2. The third row $101 \in \mathcal{L}_{G_P}$ because $S \to LR \to L\,\overrightarrow{B}\,R \to L\,\overrightarrow{B}\overleftarrow{B}\,R \to 1\,\overrightarrow{B}\overleftarrow{B}\,R \to 10R \to 101$. Later, we will make use of a subclass of unrestricted grammars equivalent to blocked cellular automata, which generalize the example and which are still capable of generating all recursively enumerable languages; that is, they are *universal*. A surprising consequence of universality is that the membership question for recursively enumerable languages is sometimes impossible to answer!

2.2 DNA Complexes and Self-assembly Rules

Grammars turn out to have a close relationship to the self-assembly models we discuss here. However, to make this relationship precise, we define our model formally.

A *DNA complex*[2] is a connected directed graph with vertices labeled from $\{A, C, G, T\}$, edges labeled from $\{backbone, basepair\}$, with at most one incoming and one outgoing edge of each type at each node (thus at most four incident edges total), and where for every basepair edge $x \to y$ there is a reciprocal basepair edge $y \to x$. Furthermore, all base-pairing in a DNA complex must be *Watson-Crick*, that is, every basepair edge must be within a subgraph isomorphic to one of the 10 given in Figure 1a.

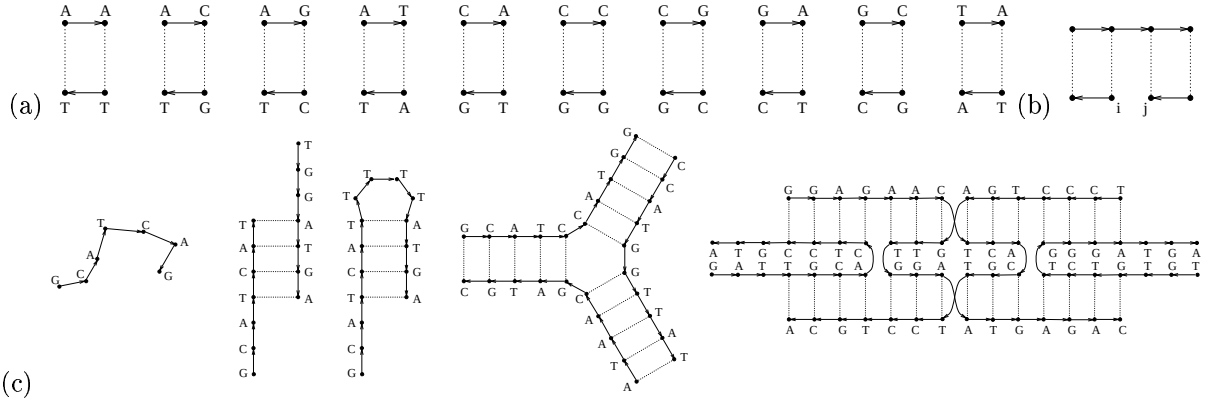

Figure 1: Some DNA complexes. Solid lines represent backbone edges; each dotted line represents a pair of reciprocal basepair edges. (a) The 10 Watson-Crick subgraphs. (b) The valid ligation site. (c) A strand, a duplex with sticky ends, a hairpin with a sticky end, a 3-armed branched junction, and a DAO double crossover (DX) unit with sticky ends.

A DNA complex (just *complex* for short) represents several DNA polynucleotides bound together by Watson-Crick hybridization. Note that this representation supports a rich variety of DNA structures, but structures such as triple helices are missing; similarly, it is lacking notions of geometry and topological linking. Also, we must be careful because it is possibly to specify physically impossibly structures.

It will be useful to introduce a few examples of DNA complexes, shown in Figure 1c. A *strand* consists of a chain of backbone-connected nodes, with no basepair edges. Strands may be either linear or circular. A *duplex* consists of two strands with contiguous basepair edges between them. A duplex may optionally have a *sticky-end* on either end. An *n-armed (branched) junction* consists of n duplex arms arranged around a central point. A *double crossover unit* (DX unit) consists of two adjacent duplexes with two points of strand exchange[3]. For formal reasons, the empty graph ϵ is a DNA complex.

We now define some operations on complexes. In our model, hybridization is indicated by $C_1 +_B C_2 = C_3$, where $+_B$ denotes the formation of basepair edges B between nodes of C_1 and nodes of C_2. If the graph consisting of both C_1 and C_2 and the edges B is a DNA complex, then C_3 is that graph; else $C_3 = \epsilon$ (for example, if a new edge joins two T's). The hybridization operation will be used to describe self-assembly, below.

[2] Similar to Beaver's *cluster* [Beaver].

[3] Real DX molecules [Fu] come in a number of geometric varieties (we use "DAO" here), each of which put constraints on the symmetry and the number of nucleotides between crossover points. We ignore these constraints in the theoretical section.

To analyze the complexes present after self-assembly, we introduce two other operations based on ligation and denaturing. $C' = ligate(C)$ is obtained by adding a backbone edge from node j to node i in every occurrence of the subgraph shown in Figure 1b, so long as nodes i and j have no other incident backbone edges.

To model the denaturing of a complex, we define $\{C_i\} = denature(C)$ to be the set of all *strands* in C, i.e., each C_i is a backbone-connected component of C (with no basepair edges). Note that if C contains topologically linked circular strands, then *denature* will "magically" unlink them from each other[4].

In analogy to formal language theory, we define a *language of DNA complexes* to be a well-defined, possibly infinite set of DNA complexes. We can generate a language of complexes $\mathcal{L}_{R,A}$ by applying *self-assembly rules* R to an initial language A, usually finite[5]. The rules R specify which hybridizations $C_1 +_B C_2 = C_3$ are allowed. Let $\hat{\mathcal{L}}_{R,A}$ be the transitive closure of A under all allowed hybridizations. In other words, (a) $A \subset \hat{\mathcal{L}}_{R,A}$, (b) if $C_1, C_2 \in \hat{\mathcal{L}}_{R,A}$ and $C_1 +_B C_2 = C_3$ is allowed, then $C_3 \in \hat{\mathcal{L}}_{R,A}$, and (c) no other complexes are in $\hat{\mathcal{L}}_{R,A}$. Now let $\mathcal{L}_{R,A} \subset \hat{\mathcal{L}}_{R,A}$ consist of those complexes for which no further hybridization is allowed. Loosely, $\mathcal{L}_{R,A}$ is meant to model the DNA structures which would form given an infinite volume of DNA and infinite time, presuming that only the hybridizations allowed by R are physically relevant, and ignoring transient structures.

We will be especially interested in the self-assembly rules[6] R_1^T which allow $C_1 +_B C_2 = C_3 \neq \epsilon$ iff (1) the subgraph of C_3 induced by B contains exactly two T-mer (or longer) strands and (2) at most two edges lead to or exit from this subgraph. Thus, R_1^T allows only hybridization of sufficiently long sticky-ends, as illustrated in Figure 2.

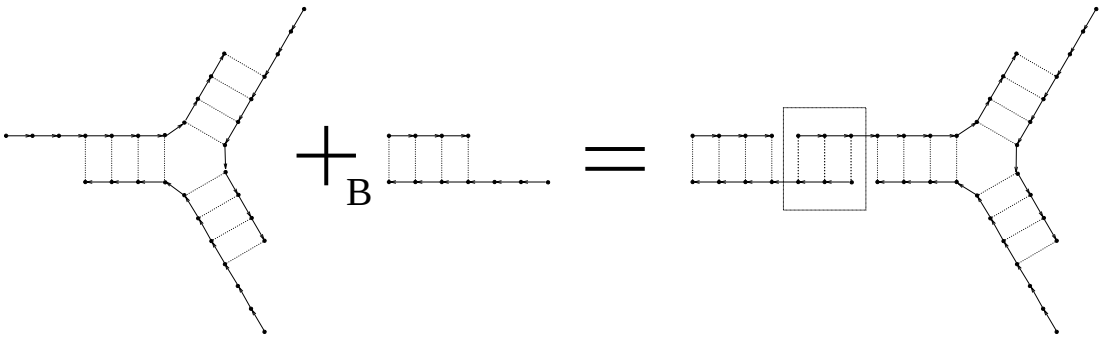

Figure 2: A hybridization $C_1 +_B C_2 = C_3$ allowed by R_1^3. The edges of B are emphasized in C_3, and the subgraph induced by B has a dotted box around it.

Both *ligate* and *denature* can be generalized to set operations by applying the operation to each complex in the original set, and taking the union of all complexes that result. Since single-stranded DNA can be identified with its sequence, written 5' $\rightarrow$ 3', we can consider *denature* to be a function from sets of complexes to sets of strings over $\{A, C, G, T, \cdot\}$, where $\cdot$ is used to indicate a circular DNA strand.

Finally, we note that to represent strings in alphabets Σ other than $\{A, C, G, T\}$, we may use a *prefix-free codebook* $\mathcal{C}$ which assigns to each symbol σ in Σ a string $\mathcal{C}_\sigma$ over $\{A, C, G, T\}$

[4]Circular strands are not necessary in our constructions, but they must be considered in Theorem 2(2).
[5]Logicians may think of A as "axioms" while R may be thought of as "inference rules".
[6]The subscript "1" is used because these rules give rise to essentially one-dimensional complexes.

such that no string is a prefix of another string. A DNA sequence $s = s_1 s_2 \ldots s_n$ can then be translated into a string $\mathcal{C}(s)$ over Σ by scanning through s from left to right: if s_i begins a subsequence of s which exactly matches some $\mathcal{C}_\sigma$, then s_i is replaced by σ, else s_i is erased; then s_{i+1} is processed, and so forth. For example, if $\Sigma = \{0, 1\}$, $\mathcal{C}_0 = CAG$, and $\mathcal{C}_1 = CTC$, then $\mathcal{C}(AAACTCTCAGTCAG) = 1100$.

In summary, given a finite set of complexes A, self-assembly rules R, and codebook $\mathcal{C}$, we can obtain a language of complexes $\mathcal{L}_{R,A}$ as well as a language of strings $\mathcal{L}_{R,A,\mathcal{C}} = \mathcal{C}(denature(ligate(\mathcal{L}_{R,A})))$.

We now turn to our results. The theorems are stated, explained, and examples are given. Full proofs will appear elsewhere.

2.3 Linear Self-assembly is equivalent to Regular Languages

In this section we address the question of what can be computed by the self-assembly DNA which obeys Properties (1-4), (5a), and (6a) or (6b). This is the familiar case of the self-assembly of long duplex DNA from many small oligonucleotides or sticky-ended fragments. That is, self-assembly begins with oligonucleotides or duplex DNA with sticky ends, and proceeds at a constant temperature, allowing only permanent binary events with a single perfectly complementary hybridization site and no intramolecular hybridization. We make this question precise in our model by asking, what languages of strings $\mathcal{L}$ can be achieved as $\mathcal{L}_{R_1^T,A,\mathcal{C}}$ for some choice of T, $\mathcal{C}$, and A where A contains only linear duplex complexes?

The following[7] can be proved by construction:

Theorem 1. (1) For all regular languages $\mathcal{L}$, there exists a positive integer T, a codebook $\mathcal{C}$, and a set of linear duplexes A such that $\mathcal{L} = \mathcal{L}_{R_1^T,A,\mathcal{C}}$. (2) For all positive integers T, codebooks $\mathcal{C}$, and sets of linear duplexes A, $\mathcal{L}_{R_1^T,A,\mathcal{C}}$ is a regular language.

We will sketch the construction used in the proof of (1) – see Figure 3 for an example. Consider a regular grammar G for $\mathcal{L}$. We design sufficiently dissimilar sequences S_i (we call their Watson-Crick complements S_i') for all the terminal and nonterminal symbols in G. For each rule $A \to p_1 \ldots p_n B$, we design a duplex with a sticky end S_A', and internal duplex region $S_{p_1} \ldots S_{p_n}$, and a sticky end S_B if B is present. We also design a duplex with one blunt end and a sticky end S_S, to represent the start symbol S. These duplexes make up the initial set of complexes A. T is chosen to be the length of the nonterminal sequences S_i. After self-assembly, the terminal complexes in $\mathcal{L}_{R_1^T,A}$ will correspond to derivations in G. After ligation, each complex will be a blunt-ended duplex whose sequence consists of terminal sequences interspersed with nonterminal sequences. A codebook with $\mathcal{C}_i = S_i$ for each terminal symbol i will "erase" the nonterminal sequences; thus $\mathcal{L}_{R_1^T,A,\mathcal{C}}$ will be exactly $\mathcal{L}$. □

A sketch of the proof of (2) is as follows: we construct a regular grammar G which generates exactly the strands in $denature(ligate(\mathcal{L}_{R_1^T,A}))$. This requires creating a nonterminal symbol for each sticky end of a duplex in A, and considering all (finitely many) T-or-more base overlaps of these sticky-ends; a grammar rule is provided for each such interaction. Care must be taken for gaps and for sticky ends which have no interactions – both lead to termination

[7] We note that this theorem still holds when "duplexes" is replaced by "strands".

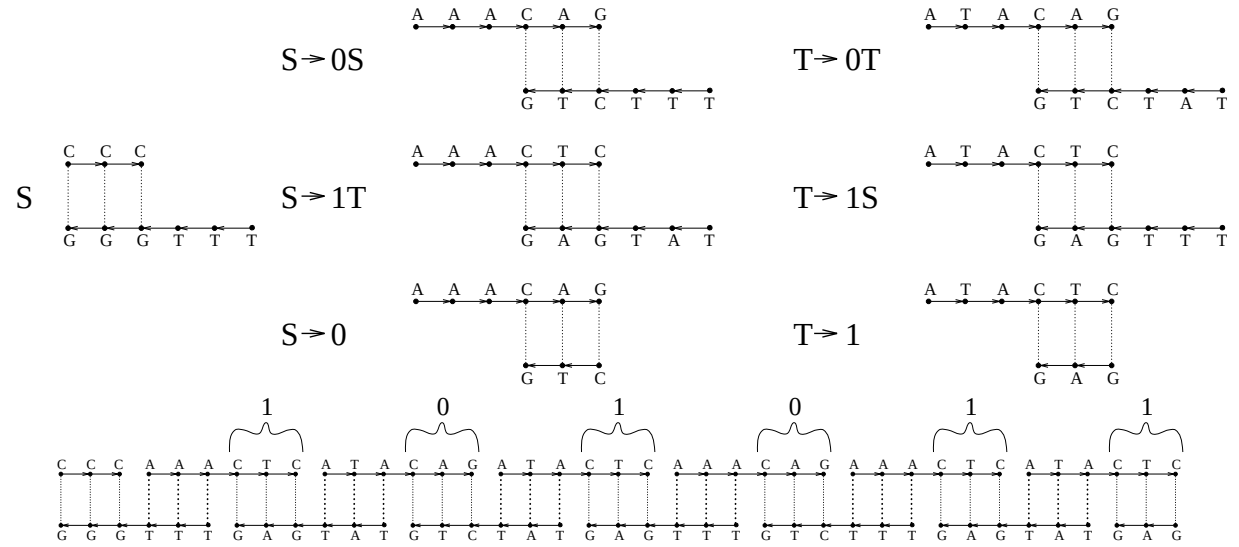

Figure 3: The initial complexes A corresponding to the regular grammar G_E, and an example derivation. Note that the self-assembly of the derivation could have occurred in any order. Subsequent ligation and denaturing will produce two strands (top and bottom) from this terminal complex. The codebook $\mathcal{C}$ defines $\mathcal{C}_0 = CAG$ and $\mathcal{C}_1 = CTC$. We use R_1^3.

of the strand sequence, and may require a rule using the start symbol S. Translation by the codebook can be effected by applying a nondeterministic finite state transducer [Ginsburg] to $\mathcal{L}_G$, yielding a regular language equal to $\mathcal{L}_{R_1^T,A,\mathcal{C}}$. □

Thus, our model for linear self-assembly does not permit very interesting computations. It should be emphasized that simple extensions might allow for more complex computations. For example, suppose hairpins appear in A in addition to duplexes. Then, for example, we could replace the duplex for S (Figure 3) by the hairpin  , and change the codes for 0 and 1 to the Watson-Crick palindromes $CCGG$ and $CGCG$. Now both the top and bottom strands code the 0 and 1 sequences; furthermore, after ligation the top and bottom strands are joined together by the hairpin. Consequently, we generate the set of all palindromes in which the number of ones is a multiple of four – which is not a regular language! How far can we push this idea?

2.4 Dendrimer Self-assembly is equivalent to Context-free Languages

It has been observed [Abrahams-Gessel] that dendrimer self-assembly looks formally identical to context-free grammars. This observation translates very nicely into DNA self-assembly of branched junctions into tree-like complexes. Therefore, in this section we address the question of what can be computed by the self-assembly of DNA which obeys Properties (1-4), (5a), and (6b-d). That is, self-assembly begins with duplexes, hairpins, and 3-armed junctions with sticky ends, and proceeds at a constant temperature, allowing only permanent binary events with a single perfectly complementary hybridization site and no intramolecular hybridization. We note that this form of self-assembly has not been widely studied in the lab, and that full self-assembly would be limited not only by material but also by geometric

(steric) interference and volumetric constraints[8]. Nonetheless, our abstract model allows us to ask the following precise question: what languages of strings $\mathcal{L}$ can be achieved as $\mathcal{L}_{R_1^T,A,\mathcal{C}}$ for some choice of T, $\mathcal{C}$, and A where A contains only duplexes, hairpins, and 3-armed junctions?

An extra complication that immediately arises is the possibility that circular strands may form. Recall our convention that *denature* returns "dotted" sequences to represent circular strands, but didn't specify which permutation of the circle to use. It becomes convenient to work with equivalence classes of sequences, where $\cdot S \cong \cdot T$ if the sequences S and T are circular permutations of one another. Languages $\mathcal{L}_1$ and $\mathcal{L}_2$ are deemed *equivalent* if for every sequence S in one language, there is an identical or equivalent sequence T in the other language.

The following[9] can be proved by construction:

Theorem 2. (1) For all context-free languages $\mathcal{L}$, there exists a positive integer T, a codebook $\mathcal{C}$, and a set of duplexes, hairpins, and 3-armed junctions A such that $\mathcal{L} = \mathcal{L}_{R_1^T,A,\mathcal{C}}$. (2) For all positive integers T, codebooks $\mathcal{C}$, and sets of duplexes, hairpins, and 3-armed junctions A, $\mathcal{L}_{R_1^T,A,\mathcal{C}}$ is equivalent to a context-free language.

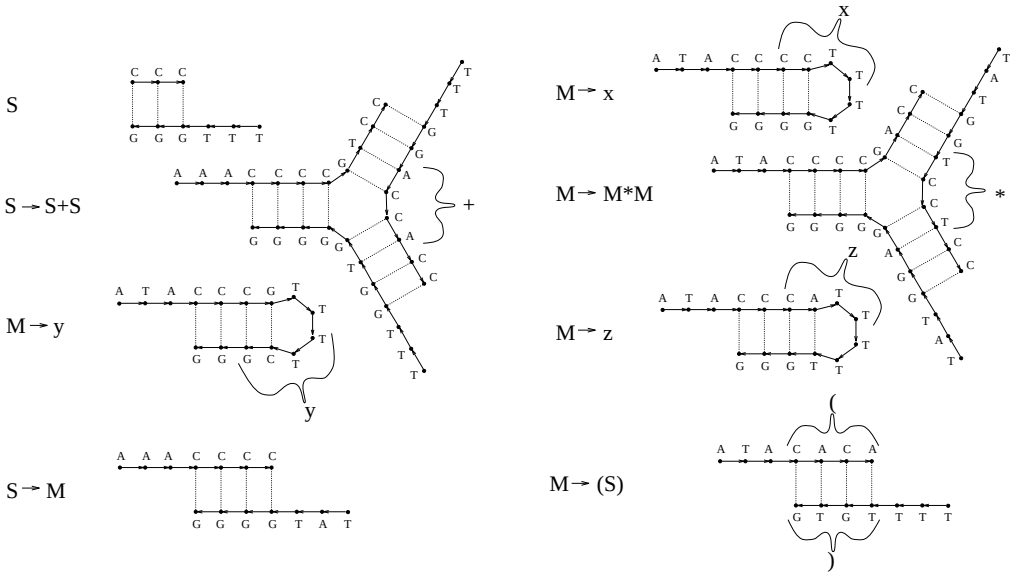

Figure 4: The initial complexes A corresponding to the regular grammar G_F. The codebook $\mathcal{C}$ defines $\mathcal{C}_x = CCTT, \mathcal{C}_y = TTCG, \mathcal{C}_z = CATT, \mathcal{C}_(= CACA, \mathcal{C}_) = TGTG, \mathcal{C}_+ = ACCA$, and $\mathcal{C}_* = TCCT$. We use R_1^3.

We will sketch the construction used in the proof of (1) – see Figures 4 and 5 for an example. The construction is similar to that in Theorem 1. Consider a context-free grammar G for $\mathcal{L}$. Note that there is an equivalent grammar $\hat{G}$ which uses rewriting rules of the form $A \to pBqCr$ where p, q, and r are (possibly null) strings of terminal symbols, and A, B, and C are single nonterminal symbols (or null). Again, we design sufficiently dissimilar sequences S_i for all the terminal and nonterminal symbols used in $\hat{G}$. For rules of the form

[8]Consider a tree which branches at every opportunity. It has 2^n nodes within n steps of the center; but the volume of space within n steps grows only as n^3.

[9]We note that this theorem still holds when "duplexes, hairpins, and 3-armed junctions" is replaced by simply "complexes". That is to say, this is a fully general theorem for self-assembly under R_1^T.

$A \to pB$ or $A \to Bp$ (B not null), we design a duplex as before. For rules of the form $A \to p$, we design a hairpin with the sequences for p in the stem. We design a 3-armed junction for each rule of the form $A \to pBqCr$ (B and C not null); it has sticky ends for S'_A, S_B, and S_C, and the sequences for p, q, and r are placed on the arms. As before, we design a blunt-ended duplex for the start symbol S. These complexes make up the initial set of complexes A. As before, at the appropriate "temperature" T, the terminal complexes will correspond to derivations in $\hat{G}$, and ligation will convert each complex into a single strand which encodes the derivation. Processing with the codebook for the terminal symbols will "erase" the nonterminal sequences, and $\mathcal{L}_{R_1^T,A,\mathcal{C}}$ will be exactly $\mathcal{L}$. □

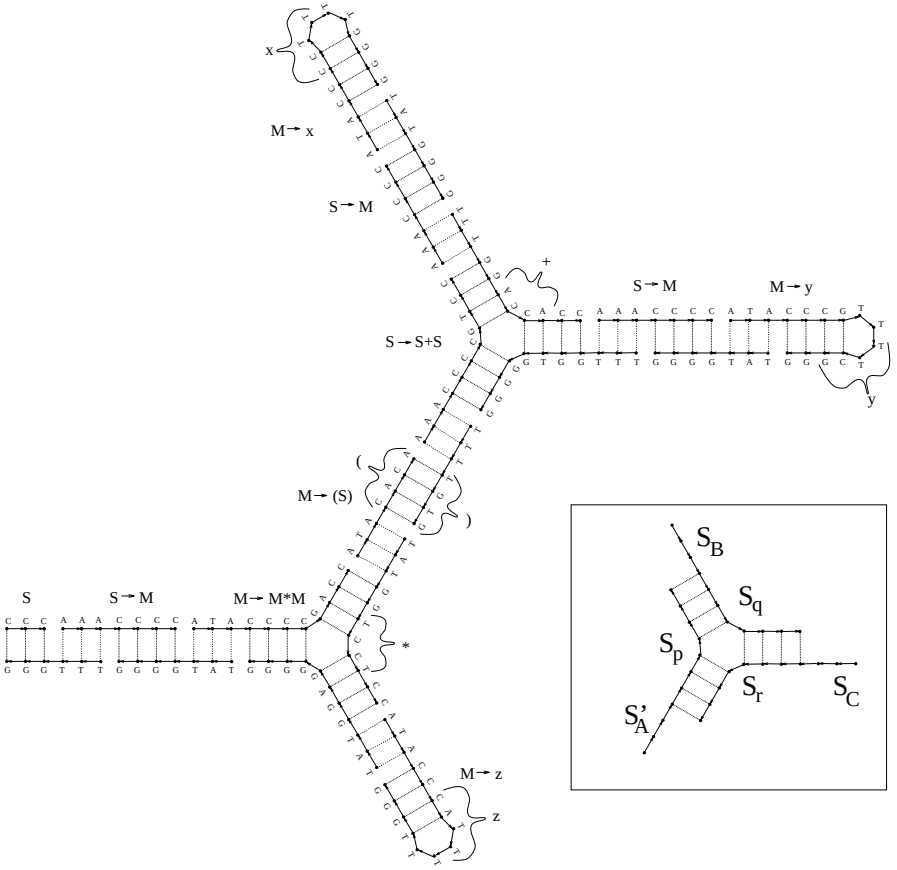

Figure 5: An example derivation by self-assembly of the complexes A corresponding to the regular grammar G_F. Note that the self-assembly of the derivation could have occurred in any order, using R_1^3. Subsequent ligation will produce a single strand from this terminal complex. Inset: the three-armed junction corresponding to the generic rewriting rule $A \to pBqCr$.

The proof of (2) also follows the form of the proof of Theorem 1, only now we construct a context-free grammar G which, loosely speaking, generates sequences corresponding to backbone paths through complexes in $ligate(\mathcal{L}_{R_1^T,A})$, where gaps are filled in with the symbol $\diamond$, and where several (but not necessarily all) permutations of each circular strand are given using $\cdot$. This language is then passed through a nondeterministic transducer which returns the strand sequences in $\{A, C, G, T\}$ and circular strand sequences in $\{A, C, G, T, \cdot\}$. As before, the final strings are produced by another nondeterministic transducer, which this time translates using the codebook. Thus the final language is context-free, and is equivalent to $\mathcal{L}_{R_1^T,A,\mathcal{C}}$. □

More intuitively, we can reason that because no intramolecular hybridizations are allowed by R_1^T, the initial complexes can aggregate only into tree-like structures. No matter how convoluted the original complexes are, paths through the resulting tree-like structures are well modeled by context-free languages.

Our model of self-assembly of DNA into tree-like structures has strictly more computational power than the model of linear self-assembly. However, it is still a far cry from universal computation. It turns out that when we attempt to model intramolecular interactions, in the form of cooperative binding sites, a much more powerful model results. We consider a particular case in the following section.

2.5 Two Dimensional Self-assembly is Universal

To prove that two dimensional self-assembly can be universal, it suffices to demonstrate a restricted class which is universal. We review the class of structures introduced in [Winfree], which are geometrically based on a lattice of double crossover (DX) units of DAO type [Fu]. It was shown in [Winfree] that the self-assembly of DX units can directly mimic the operation of an arbitrary one dimensional cellular automata system. An example is shown in Figure 6, where a simple blocked cellular automaton rule (corresponding to the unrestricted grammar G_P of Section 2.1, but without the termination rules) is used to generate a Sierpinski triangle pattern.

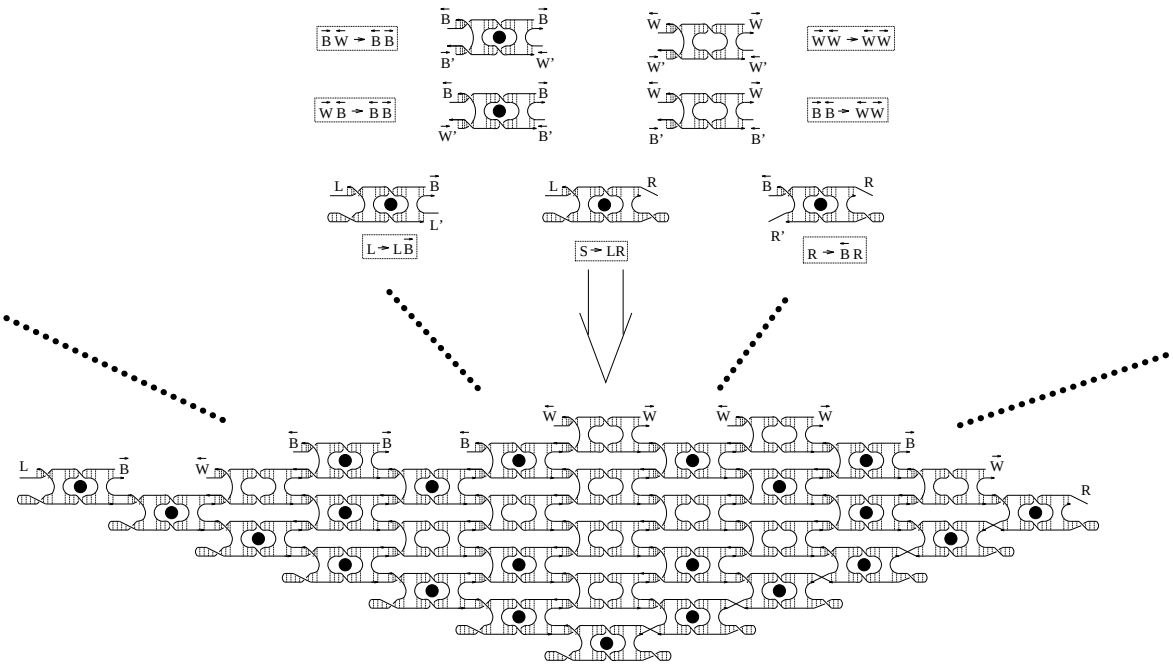

Figure 6: An algorithmic pattern in a self-assembled lattice. At the top, the seven initial DX units in A are shown (the black dot is a visual aid to identify "black" complexes), involving 22 oligonucleotides. The corresponding rewriting rules from G_P are presented in boxes. The units use 12 unique sticky end sequences, denoted by $\{L, R, \overleftarrow{B}, \overrightarrow{B}, \overleftarrow{W}, \overrightarrow{W}\}$ and their complements L' etc. The L and R sequences are both length T; the other sequences are length $T/2$. Upon self-assembly according to R_2^T, a V-shaped chain of the lower three units is formed due to hybridization of L and R, while the open slots in the initial chain are filled by the unique unit whose sticky ends match those on both sides of the slot. In this example, the process continues indefinitely. Each strand in $ligate(\hat{\mathcal{L}}_{R_2^T,A})$ represents one or two columns of Pascal's triangle mod 2.

The model of self-assembly used here follows Properties (1-4), (5b), and (6e), and it is motivated by additional physical concerns. As shown in Figure 6, the hybridization events may now involve *two* binding sites arranged as a *slot*. Geometry becomes important; only sticky ends which are close to each other and arranged properly may form a slot where binding can occur. Physically, one sticky end of an unattached DX unit would hybridize to one side of the slot, followed shortly by (the now intramolecular) hybridization of the DX unit's other sticky end to the slot's other binding site. For full computational generality, it is critical that a DX unit which matches one site in a slot, but not the other site, will *not* hybridize to the lattice. Under appropriate conditions, DX units which bind to only one site in a slot would soon dissociate, while fully matching DX units would bind nearly irreversibly. We therefore model slot-filling as a single permanent binary event involving two binding regions, and T is chosen so that single-site binding will not occur.

We emphasize that this form of DNA self-assembly has not yet been demonstrated experimentally, although we report some preliminary results in Section 3.

We must define new self-assembly rules: R_2^T allows hybridizations allowed by R_1^T, and additionally allows two-region slot-filling hybridizations between complexes containing the subgraphs shown in Figure 7, so long as the total number of basepair edges in B is at least T. This rule is meant to model local geometry in complexes; it will be a good model only for certain structures, including (we believe) the ones used in our construction.

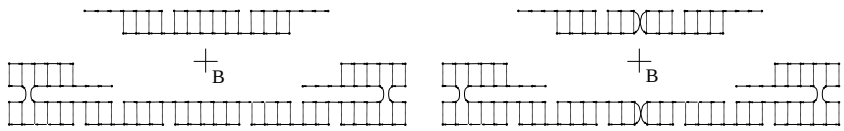

Figure 7: Two allowed slot-filling hybridizations in R_2^6. These graphs represent requires subgraphs of the complexes C_1 and C_2 in $C_1 +_B C_2 = C_3$. Other positions of nicks are also allowed, as are other lengths of the duplex regions.

The following can be proved by construction:

Theorem 3. (1) For all recursively enumerable languages $\mathcal{L}$, there exists a positive integer T, a codebook $\mathcal{C}$, and a set of duplexes and DX units A such that $\mathcal{L} = \mathcal{L}_{R_1^T,A,\mathcal{C}}$. (2) For all positive integers T, codebooks $\mathcal{C}$, and sets of duplexes and DX units A, $\mathcal{L}_{R_1^T,A,\mathcal{C}}$ is equivalent to a recursively enumerable language.

The proof of (1) is based on the constructions in [Winfree]. As cellular automata are capable of universal computation, for example by directly simulating Turing machines, we conclude that two dimensional self-assembly is universal. (2) follows because there is an algorithm for generating all the complexes in $\mathcal{L}_{R,A}$ so long as R is computable: keep trying new hybridizations of complexes known to be in the language, and remember the resulting complex. □

Although universal, one dimensional cellular automata are not often a convenient model for computing functions of interest, although they are faster and more efficient than 1-tape Turing Machines, due to their parallelism.

2.6 Solving the Hamiltonian Path Problem

As a concrete example of using two dimensional self-assembly for computation, we will solve the same Hamiltonian Path Problem (HPP) used in [Adleman]. Recall that the problem is to find a path from node 1 to node N which visits every node in G exactly once. Our algorithms for solving HPP will be based on:

1. Generate all paths from node 1 to node N.
2. In each path, sort the vertices into increasing order.
3. For each path, check that the result is exactly "$1, 2, 3, \ldots, N$".
4. Output any path which passes the test, if one exists.

In a preparatory step, DNA sequences are designed for the given graph and synthesized. Steps 1-3 will occur as a single self-assembly step, while Step 4 consists of sequencing circular DNA of known length.

For the graph used in [Adleman] (shown in Figure 8a), $N = 7$ and we will require a total of 68 DX units of DAE type. Shown in Figure 8b, units 1 through 20 are responsible for Step 1

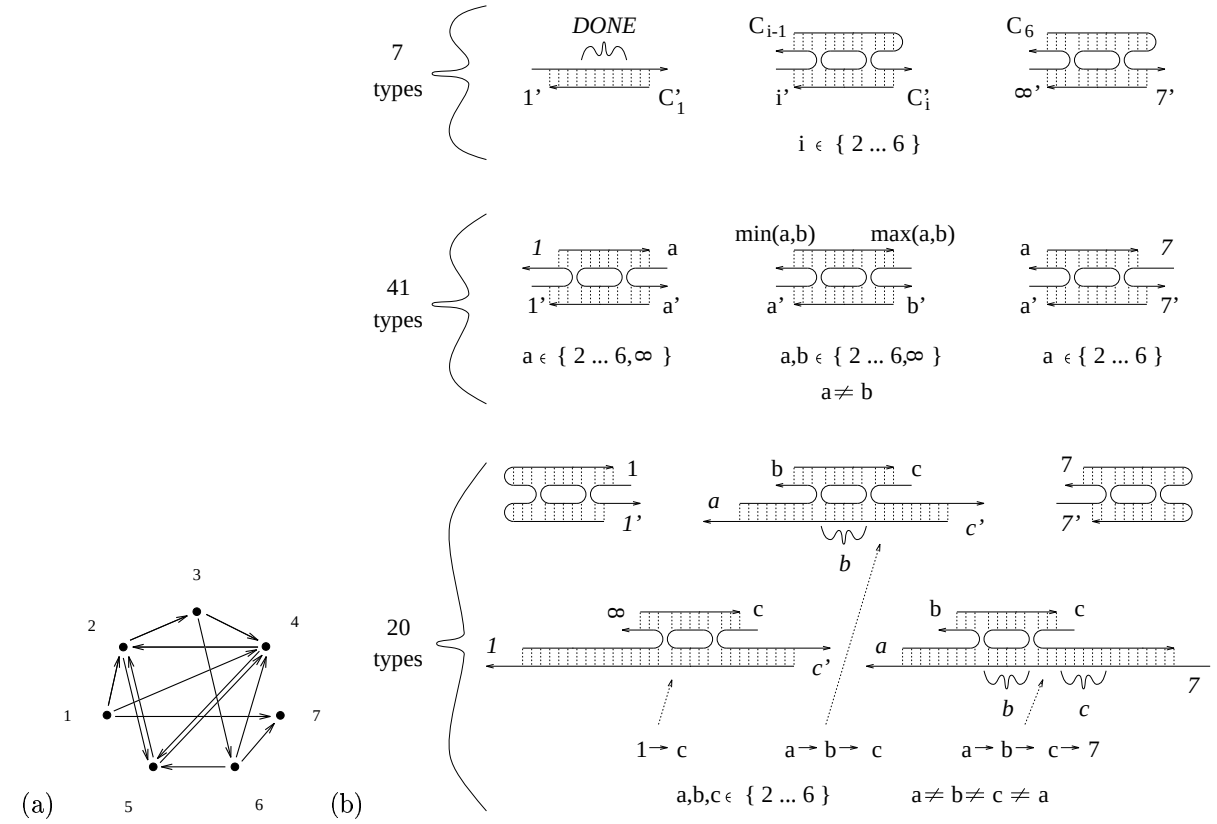

Figure 8: (a) The 7 node graph G. (b) Rule molecules (DAE type) for solving the Hamiltonian Path problem. Sticky ends of length $T/2$ are $\{1, 2, 3, 4, 5, 6, 7, \infty, C_1, C_2, C_3, C_4, C_5, C_6\}$ and their complements; sticky ends of length T are $\{1, 2, 3, 4, 5, 6, 7\}$ and their complements. $DONE$ is a special sequence which indicates completion of a lattice. The variables i, a, b, c are used to concisely represent a multiplicity of rule molecules; italicized variables indicate length T sticky ends. For example, in the lower set, 15 units are designed from the central schema, one for each pair of incident paths $a \to b, b \to c$.

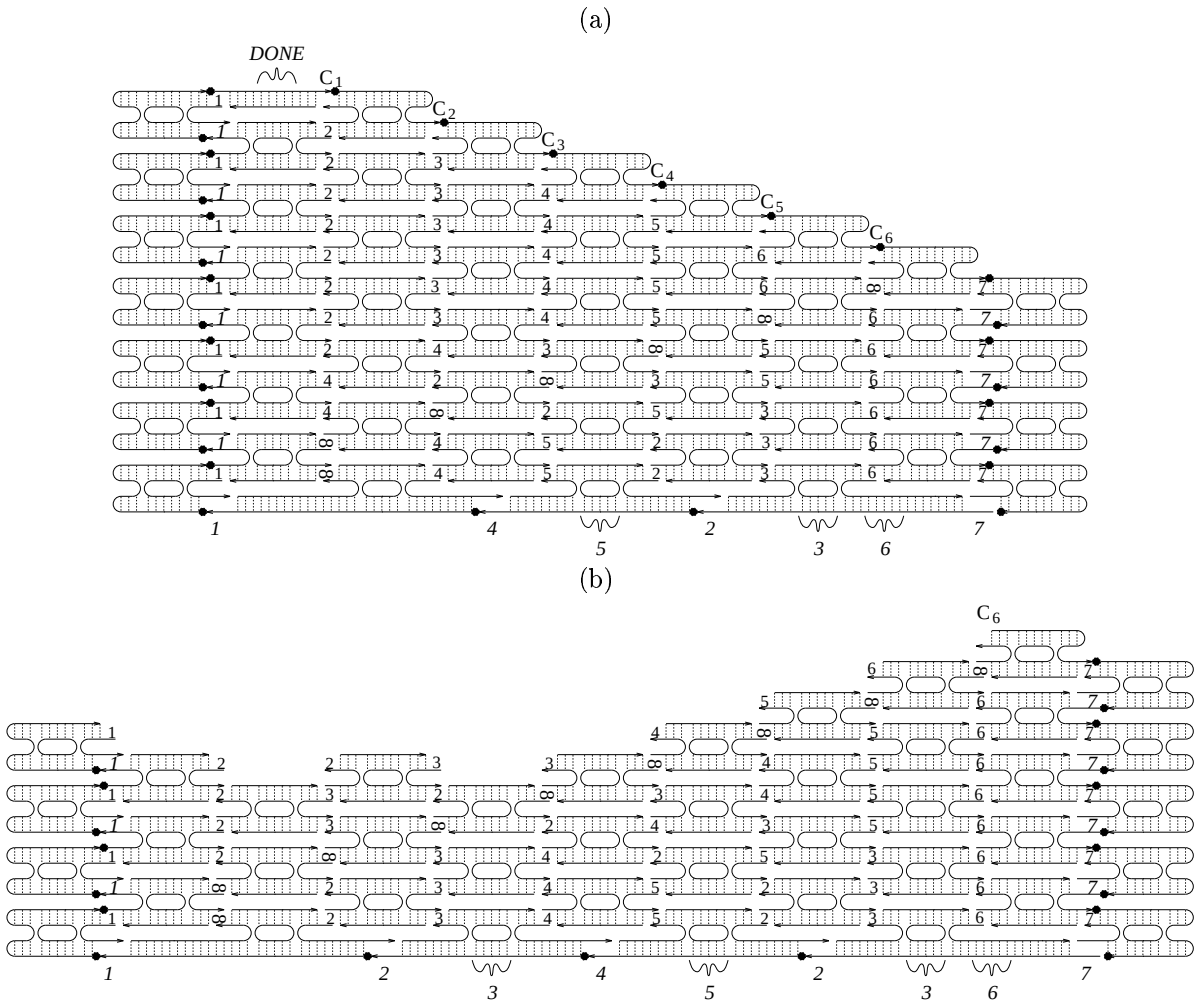

Figure 9: Terminal complexes after annealling. Black dots show nicks which will be ligated. (a) The lattice verifying the Hamiltonian path 1452367. (b) The lattice rejecting the invalid path 123452367.

of the algorithm (the bottom layer in Figure 9a,b); these units are analogous to the oligos in Adleman's solution[10]. Units 19 through 61 are responsible for Step 2 of the algorithm; sorting is accomplished by the Odd-Even Transposition Sort [Knuth]. When the symbol ∞ has travelled all the way to the right, the sorting is complete and Step 3 is initiated, using units 62 through 68.

Each terminal complex either (a) encodes a valid Hamiltonian Path, in which case the complex is complete (Figure 9a), and ligation cyclizes the outer ring, but not the inner ring[11]; or (b) encodes an invalid path, in which case the terminal complex contains unfilled, open slots (Figure 9b) and will produce no cyclic strands when ligated[12]. Thus Step 4 can be achieved by separating cyclic from linear DNA strands (e.g. by 2D gel electrophoresis,

[10] Adleman's oligos encoded individual edges in the graph, whereas ours encode pairs of edges. Also, knowing that a Hamiltonian path in this graph must visit exactly 7 nodes, our units are devised such that only odd-length paths can form completely.

[11] This can be ensured either by leaving an unmatched base on the sticky ends for interior units, or by phosphorylating only units which occur on the outer edge.

[12] Note that if a path visits a node twice, there will be a gap in the "Step 2" portion of the terminal complex; if a path fails to visit some node, there will be a gap in the "Step 3" portion of the terminal complex.

by exonuclease digestion, or by affinity purification based on the $DONE$ sequence) followed by amplification and sequencing.

Let us briefly compare this molecular algorithm to the one used in [Adleman]. To solve a graph with N nodes and E edges, Adleman used roughly $N + E$ oligos and N laboratory steps. We would use roughly $E^2/N + N^2 + N$ DX units (each requiring up to 5 strands) and a constant number of laboratory steps (synthesis, annealing, sequencing)[13].

Because two dimensional self-assembly can simulate arbitrary cellular automata, similar algorithms can be designed for any computational purpose. For example, an N-variable size s Circuit-SAT problem can be solved using roughly Ns DX units and a constant number of laboratory steps after synthesis.

2.7 Three Dimensional Self-assembly Augments Computational Power

A trivial corollary of the universality of two-dimensional self-assembly is that if three dimensional structures are allowed, self-assembly is still universal. It is of greater interest if we can exploit all three dimensions to allow for more efficient or more reliable computations. We propose a scheme to do exactly that, again for concreteness using DAO units as our basic building block. In this section, we will present some physical considerations, but we will not formally define R_3^T.

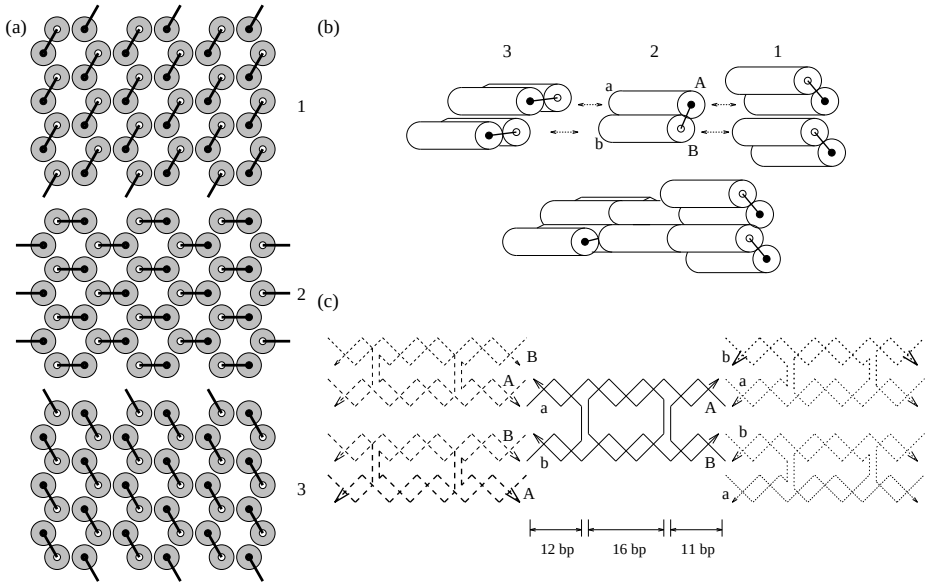

Figure 10: Plan for three dimensional lattice. (a) Three cross-sections through the final lattice, corresponding to the three sections indicated in (b). Circles represent cross-sections through a double helix of DNA; bars indicate which helices are part of the same DAO unit. (b) Relative angles of five DAO units are indicated. For perfect 120° angles, helical twist between 31.5 and 35.5 °/ bp is required. (c) Detail of the DAO units. A single DAO unit, with sticky end a complementary to A and b complementary to B, suffices to generate the entire lattice. For computations, the sticky ends are indexed s.t. a_i binds only A_i and b_i binds B_i.

[13]How feasible these imagined laboratory steps would be is, of course, an open question. However, once the laboratory techniques have been debugged, conceivably our algorithm could be carried out in a single day's work – regardless of the size of the graph (volume permitting). A concern is that, moreso than in Adleman's algorithm, the success of our algorithm is critically dependent upon ligation yields. For example, if ligation is 80% effective, then only $0.8^{30} = 0.1\%$ of the correct terminal complexes will be fully cyclized in our $N = 7$ graph. Also, since each path requires a DNA molecule roughly 100 times larger than the DNA used in Adleman's algorithm, greater reaction volumes will be necessary.

We begin by noting that the solid angle between two adjacent DAO units is determined by the length of the linker arm between them. For the planar lattice, we choose a length such that the angle is approximately 180°. Alternatively, we can choose lengths such that the angle is near 120°, the appropriate value for a "honeycomb lattice" as shown in Figure 10.

As in the case of the two dimensional lattice of DAO units, computation is brought about by judicious choice of the sticky end sequences on several DAO units. The three dimensional lattice thus formed is equivalent to the space-time history of a 2D blocked cellular automata.

The particular incarnation of three dimensional lattice chosen here is clearly not unique, and it is suggested more as a brain-teaser than as a serious proposal; other geometries are possible, perhaps having preferable practical characteristics.

2.8 Discussion

We have analyzed the computational power of three different regimes of self-assembly in our abstract model, and we have speculated on an extension into the self-assembly of a three dimensional lattice.

The essential construction in the linear case is due to [Adleman] who used it to construct paths through graphs. [Boneh] and [Winfree] observed that linear self-assembly is capable of generating regular languages. Here, we state the result in the context of our formal model, and we show that linear self-assembly of duplexes is *limited* to regular languages. This point requires making the distinction between self-assembly processes with and without hairpins, as shown by the palindromes example. Linear self-assembly has been exploited in many laboratory experiments – both by molecular biologists and by people interested in molecular computation – and although its intricacies are not completely understood, there is a wide foundation of practical experience.

The self-assembly of branched junctions into dendrimer structures seems to be a relatively unexplored idea. For example, in [Ma] it is observed that identical three-armed junctions with two complementary sticky ends can cyclize. If cyclization cannot be prevented, many context-free grammars would be impossible to implement by self-assembly. Another concern comes from geometry: if the desired tree-like structure contains too many branches, steric hindrance may prevent further associations from occurring. Thus it is not known to what extent the technique will be practical.

The self-assembly of DX units into a two-dimensional lattice is also an unconventional idea, yet to be demonstrated in the laboratory. Some first steps in this direction are reported in the next section of the paper, where the slot-filling reaction is explored.

It is interesting to observe that the Chomsky Hierarchy of languages, developed originally for the study of human languages, also arises naturally in the study of self-assembling structures. The progression from regular to context-free to recursively enumerable languages can be seen to parallel both (a) the progression from linear to dendrimer to planar lattice structures, and (b) the progression from "rule molecules" with effectively one input and one output, to those with one input and two outputs, to those with two inputs and two outputs.

One should note that all the previous arguments ignored the kinetic framework implicit in the process of annealing that we originally consider. Specifically, we expect that longer

complementary regions hybridize first. Annealing could be represented in our model as recursive computation of languages:

$$\mathcal{L}^{anneal}_{R_1,A,\mathcal{C}} = \mathcal{C}(denature(ligate(\hat{\mathcal{L}}_{R^1_1,\hat{\mathcal{L}}_{R^2_1,\hat{\mathcal{L}}_{R^3,\ldots}}})))$$

The kinetic aspects of this model of linear self-assembly may themselves be expoitable for computation. Intermolecular interactions other than the ones considered here might also provide computational advantages. Issues of concentrations and finite supply of DNA must also go into any more practical analysis.

3 Experimental Investigations

The above considerations only point to the possibility of algorithmically-patterned lattices of DNA. There are two major issues to be investigated experimentally. The first is whether homogeneous lattices will form; i.e., whether the geometric structure itself will self-assemble. The second issue is whether, in the presence of multiple units in solution, the logically correct unit will hybridize in each slot. This competitive process for filling each slot is essential for computation, as a single error can propagate throughout the entire computation. Ultimately, error rates will determine the size of lattice in which reliable computation may be performed.

We have begun an investigation of the second issue. We first build a model molecular complex, called `ABC`, which contains a single slot and no other sticky ends. `ABC` is composed of two double crossover molecules, `A` and `C`, and a duplex linker `B`. `ABC` is created by ligating eight oligonucleotides; the final structure contains four hybridized strands. Rather than test the assembly of a double crossover unit into `ABC`'s slot, we model the unit by a linear duplex "linker", called `D`. When `ABC` is properly hybridized to `D`, we call the complex `ABCD`. Completely ligated, `ABCD` is a complex catenane with four interlocked circles. To test the specificity of the hybridization, we also have a mismatched linker `D'`, which is perfectly complementary to only one of the sticky ends in the slot. We expect that `ABCD'` cannot be completely ligated, due to the mismatch, and hence `ABCD'` does not form a catenane. These molecular complexes are diagramed in Figure 11.

Experimentally, we must establish that the double crossover molecules `A` and `C` form properly upon annealing their component strands. As developed in [Fu], where the details of hybridization were probed by more extensive structural characterization, a good indication of proper association is a single band of mobility in a non-denaturing gel appropriate for the topology and molecular weight. The ligation products `ABC`, `ABCD`, and `ABCD'` (by which we mean whatever it is we get when we intend to make structures `ABC`, `ABCD`, and `ABCD'` respectively) are examined both in non-denaturing and denaturing gels; in the former we are looking for a single band of approximately the correct apparent molecular weight, while in the latter we are looking for linear strands of the lengths predicted for ligation. Ligation of double crossover molecules has previously been shown to be well-behaved [Li]. Topologically closed structures, such as `ABCD`, can be assayed by treating with an exonuclease [Ma]. Although none of these tests is absolutely rigorous, together they may give us confidence that the reactions are proceeding as predicted.

Figure 11. Sequences and Structures for ABCD.

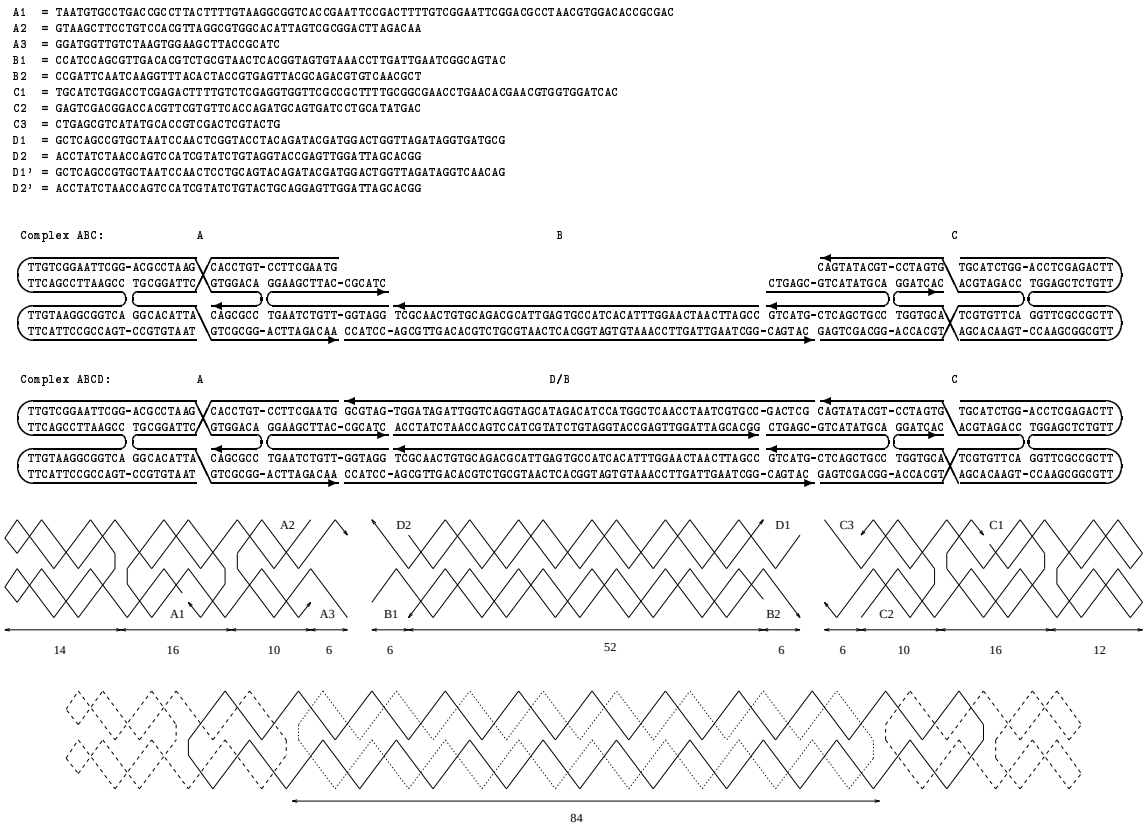

Figure 11: Sequences are written 5' to 3'. 3' ends are denoted by arrows in the diagrams, and strand labels are near 5' ends. Lengths are measured in nucleotides. Above, sequence details are given along with schematic representations of `ABC` and `ABCD` (not ligated). Below, more geometric detail is sketched for `A`, `B`, `C`, `D`, and `ABCD` (ligated). Diagrams illustrate intended structures only.

3.1 Materials and Methods

Sequence Design. The twelve strands required for `A`, `B`, `C`, `D`, and `D`' were designed by applying the principles of sequence symmetry minimization [Seeman], where the design process ensures that there are no complementary regions between strands, except as desired. In short, each double crossover molecule is designed by creating sequences appropriate for two asymmetric Holliday junctions, then juxtaposing these sequences as appropriate for a four-stranded DAO, adding hairpin sequences and re-phasing `A1` and `C1` to put the nick in the central region of the DAO. `A1`'s hairpin regions are longer than `C1`'s to allow `A1` and `C1` to be distinguished on gels. The lengths of the linkers `B` and `D` were chosen such that both DAO units should be nearly coplanar according to an estimated 10.5 base pairs per double helical full turn. Exact sequences are given in Figure 11.

Synthesis and Purification of DNA. All strands were synthesized on an Applied Biosystems 380B automatic DNA synthesizer using routine phosphoramidite procedures [Caruthers]. DNA strands were purified by denaturing polyacrylamide gel electrophoresis. DNA concentrations were estimated by OD_{260}. All strands were phosphorylated by T4 Polynucleotide Kinase (U.S. Biochemical or Promega), followed by phenol extraction and ethanol precipitation. DNA was not radiolabeled.

Formation of Hydrogen-Bonded Complexes. Complexes `A`, `B`, `C`, and `D` were formed by mixing stoichiometric quantities of each strand at concentrations near $1\mu M$ in 1x USB T4 DNA Ligase buffer (U.S. Biochemical: 66 mM Tris·HCl (pH 7.6), 6.6 mM $MgCl_2$, 10 mM DTT, 66 μM ATP; or Promega: 30 mM Tris·HCl (pH

7.8), 10 mM $MgCl_2$, 10 mM DTT, 500 μM ATP). These solutions were annealed for two hours from $80°C$ down to room temperature.

Formation of Covalently Bonded Complexes. Complexes AB, BC, ABC, ABCD, and ABCD' were formed by mixing stoichiometric quantities of annealed A, B, C, and D', followed by D after 20 minutes. Up to 50 units of T4 DNA Ligase (U.S. Biochemical or Promega) were added and solutions were incubated in a $16°C$ water bath for 2 or 8 hours. One sample of ABCD was further treated by adding $\frac{1}{10}^{th}$ volume 10x USB Exonuclease III buffer (U.S. Biochemical) and 100 units Exonuclease III (U.S. Biochemical), incubated at $37°C$ for 1 hour. Prior to being loaded in gels, solutions for gels (a) and (b) were heated to $80°C$ and again annealed to room temperature, to denature proteins and re-form hydrogen-bonded complexes. For gels (c), ligation was followed by phenol extraction and ethanol precipitation, then samples were heated to $90°C$ for 5 minutes prior to being loaded.

Denaturing Polyacrylamide Gels. Denaturing gels contain 8.3 M urea and 8% acrylamide (19:1 acrylamide:bisacrylamide). The running buffer is TBE (89 mM Tris·HCl (pH 8.0), 89 mM boric acid, 2 mM EDTA). The sample buffer contains 0.1% bromphenol blue and xylene cyanol FF tracking dyes in 80% formamide with 10 mM EDTA. Samples are heated at $80°C$ for 5 minutes immediately prior to loading. Gels are run at approximately 60 V/cm and 35 Watts, then soaked in StainsAll dye and digitized by DeskScan II on an Apple Macintosh.

Non-denaturing Polyacrylamide Gels. Non-denaturing gels contain 12.5 mM Mg^{++} and 8% acrylamide (19:1 acrylamide:bisacrylamide), 0.75 mm thick. The running buffer is TAE/Mg^{++} (40 mM Tris·HCl (pH 8.0), 20 mM acetic acid, 2 mM EDTA, 12.5 mM magnesium acetate). The loading buffer contains 0.02% bromphenol blue and xylene cyanol FF tracking dyes and 5% glycerol in ligation buffer. Gels are run at approximately 16 V/cm and 10 Watts at $4°C$, then soaked in StainsAll dye and digitized by DeskScan II on an Apple Macintosh.

3.2 Results

Formation of Complexes.

Figures 12a and 12b show several stages in the formation of ABCD. On the non-denaturing gel, the duplexes and double crossover molecules A, B, C, and D form clean bands (a, lanes 1-4) which migrate with approximately the same mobility as equivalent molecular weight duplex DNA. Ligation products AB and BC also show clean bands (a, lanes 9-10). Ligation product ABC appears as the major band in its lane (a, lane 8); another band appears at the level of AB and BC indicating incomplete ligation. Ligation product ABCD also appears, we believe, as the major band in its lane (a, lanes 7 and 5); a slower unidentified band also appears. After exonuclease treatment, the major band of ligation product ABCD is still apparent, though diminished (a, lane 6).

On the denaturing gel, we obtain further evidence of ligation activity by observing the lengths of newly created oligonucleotides. Lanes 1-4 can be used as markers for the lengths of most of the original oligonucleotides: A1 (88), A2 (52), B1 (64), B2 (52), C1 (80), C2 (52), D1 (64), D2 (52). A3 and C3, both 32 nucleotides, ran off the gel. Lane 10, product AB, shows the expected formation of A2B1 (116) and B2A3 (84); lane 9, product BC, likewise shows the formation of B1C2 (116) and C3B2 (84); and lane 8, product ABC, shows the expected formation of A2B1C2 (168) and C3B2A3 (116). Lanes 5 and 7, product ABCD, contain only three significant bands: A1 (88), C1 (80), and a band which migrates slower than a 2000 nucleotide strand, according to the marker (lane 11). This slow band is exonuclease-resistant (lane 6). We therefore conclude that the band contains the catenane A2B1C2D1:A3D2C3B2; i.e., ABCD minus the A1 and C1 loops, which apparently were not ligated. Double crossover molecules with two nicks

Figure 12. Formation and Specificity of ABCD.

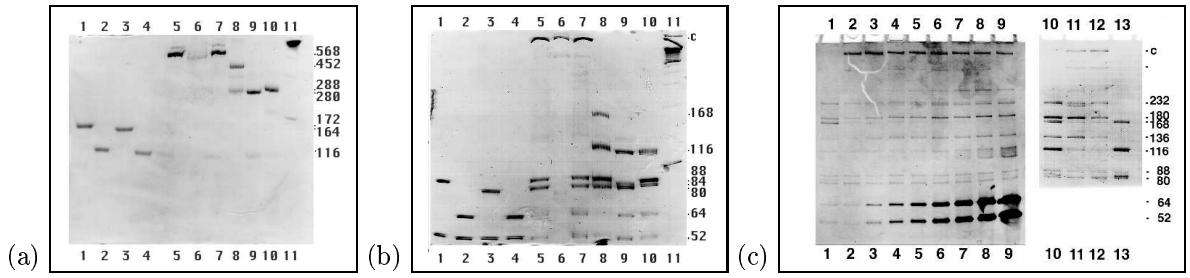

Figure 12: Numbers at right indicate estimated positions for expected products, consistent with the marker lane. (a) Non-denaturing gel electrophoresis, and (b) Denaturing gel electrophoresis. Lane 1: A (172 nucleotides). Lane 2: B (116 nucleotides). Lane 3: C (164 nucleotides). Lane 4: D (116 nucleotides). Lane 5: Product ABCD with ligase. ABCD contains 568 nucleotides. Lane 6: Product ABCD with ligase and exonuclease. Lane 7: product ABCD with ligase. Lane 8: Product ABC with ligase. ABC contains 452 nucleotides. Lane 9: Product BC with ligase. BC contains 280 nucleotides. Lane 10: Product AB with ligase. AB contains 288 nucleotides. Lane 11: 100 base-pair double-stranded ladder. (c) Denaturing gel electrophoresis. All lanes contain A:B:C in 1:1:1 stoichiometry, plus various concentrations of D:D'. Lane 1–9: D:D' = 0:1, 1:0, 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64. Lanes 10–13: D:D' = 0:20, 1:1, 1:0, 0:0. (Note: the first gel was slightly ripped during staining.)

have been shown to be stable [Zhang], suggesting that the nicks in A1 and C1 should not significantly affect the formation or stability of A or C.

Specificity of Reaction.

Figure 12c shows the results of a preliminary experiment investigating the effectiveness of D vs D' in filling the slot created by ABC. Lane 13 contains ABC, and thus has primary bands for A2B1C2 (168) and C3B2A3 (116). Lanes 2 and 12 show ligation of ABCD in 1:1:1:1 stoichiometry. The ligation apparently was not as complete as in 12b, as several bands of "partial products" are observed. The fastest band is appropriate for linear A3D2C3B2 (168) and cyclical permutations; the next band is appropriate for linear B1C2D1 or D1A2B1 (180); the next major band is appropriate for A2B1C2D1 (232) and cyclical permutations. The band below **c** is known from other gels (not shown) to be exonuclease-resistant, and the two cyclic molecule bands are thought to be an indicator of the formation of ABCD. Lanes 1 and 10 show ligation of ABC with respectively an equimolar amount or a 20-fold excess of D'. We again see the linear bands of lengths 168, 180, and 232, while bands at 136 (D2C3B2) and 116 (C3B2A3 and A3D2C3) become significant. Critically, the slow circular products are missing, suggesting that D' was only ligated on the side where it matches the slot's sticky end. Lanes 11 and 3–9 show ligation of one unit of ABC with an x-fold excess of D' and equimolar D, where x ranges from 1 to 64. In every case, the closed molecule **c** is formed, indicating that ABCD is still formed in the presence of competing D'. Additional bands also appear, possibly due to unexpected interactions involving D1 or D2.

3.3 Discussion

We interpret these results as follows. First, we believe that we are making the ABCD complex, with the caveat that we believe the nicks in A1 and C1 are not being sealed. This suggests that (1) in ABC, the linker B is properly spaced such that double crossover molecules A and C are roughly coplanar, and (2) that in each double crossover molecule, the two helical axes are also roughly coplanar. Second, we observe that D, which matches the sticky ends on both

sides of ABC's slot, outcompetes D', which matches only on one side, even when D' is 64-fold more abundant than D. We plan to quantitate the thermodynamics of this reaction in the near future.

The experiments reported here bear on the two-dimensional self-assembly process postulated in Section 2.2. These experiments are meant to model a single slot-filling step during the self-assembly of a two dimensional lattice. In our experiments, this fundamental step can occur, and with some specificity. These results encourage us to examine this step in closer detail in the future, as well as to attempt the self-assembly of an entire sheet. We hope that the self-assembly of an algorithmically patterned sheet of DNA can eventually be verified by TEM or AFM microscopy.

4 Conclusions

The self-assembly of molecules can correspond to several well known computational classes up to and including universal computation. This suggests that external processing is not an intrinsic element of molecular computation; computationally universal "one-pot" reactions seem plausible. We have shown some encouraging but preliminary experimental investigations into the fundamental computational step in our two dimensional self-assembly model. The generality of the approach used here suggests that the potential for universal computation may be widespread among self-assembly processes in nature. In addition to being interesting in its own right as a universal mechanism, it may be worth considering whether the self-assembly processes described here could be useful technologically, perhaps as part of an approach to nanotechnology [Li].

5 Acknowledgments

Erik Winfree is grateful to the many people who have helped make his foray into the world of molecules possible, enjoyable, and exciting; special thanks go to Len Adleman, Paul Rothemund, Sam Roweis, Dan Abrahams-Gessel, John Hopfield, and John Abelson who generously provided laboratory facilities at Caltech for some of the experiments reported here.

References

[Abrahams-Gessel] Dan Abrahams-Gessel (Dartmouth), personal communication, 1996.

[Adleman] Leonard Adleman. Molecular computation of solutions to combinatorial problems. *Science* 266: 1021–1024, Nov. 11, 1994.

[Beaver] Donald Beaver. Universality and Complexity of Molecular Computation. Extended abstract, available as,
`http://www.transarc.com/afs/transarc.com/public/beaver/html/research/alternative/molecute/publications/psp95.ps`

[Boneh] Dan Boneh, Chris Dunworth, Richard Lipton, and Jiri Sgall. On the Computational Power of DNA. Princeton CS Tech-Report CS-TR-499-95.
`ftp://ftp.cs.princeton.edu/reports/1995/499.ps.Z`

[Caruthers] Marvin H. Caruthers. Gene Synthesis Machines. *Science* 230: 281–285, 1985.

[Fu] Tsu-Ju Fu, and Nadrian C. Seeman. DNA Double-Crossover Molecules. *Biochemistry* 32: 3211–3220, 1993.

[Ginsburg] Seymour Ginsburg. *The Mathematical Theory of Context Free Languages*, McGraw-Hill, 1966.

[Knuth] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching (2nd ed)*, Addison-Wesley, 1973.

[Li] Xiaojun Li, Xiaoping Yang, Jing Qi, and Nadrian C. Seeman. Antiparallel DNA Double Crossover Molecules as Components for Nanoconstruction. *J. Am. Chem. Soc.* 118: 6131–6140, 1996.

[Ma] Rong-Ine Ma, Neville R. Kallenbach, Richard D. Sheardy, Mary L. Petrillo and Nadrian C. Seeman. Three-Arm Nucleic Acid Junctions Are Flexible. *Nucleic Acids Research* 14: 9745–9753, 1986.

[Seeman] Nadrian C. Seeman. Denovo Design of Sequences for Nucleic-acid Structural-Engineering. *Journal of Biomolecular Structure & Dynamics* 8 (3): 573–581, 1990.

[Winfree] Erik Winfree. On the Computational Power of DNA Annealing and Ligation. In *DNA Based Computers, Volume 27, Proceedings of a DIMACS Workshop, April 4, 1995*, ed. by E. Baum & R. Lipton, Am. Math. Soc., in press. Also available as `http://dope.caltech.edu/winfree/Papers/ligation.ps.gz`

[Zhang] Siwei Zhang, and Nadrian C. Seeman. Symmetric Holliday Junction Crossover Isomers. *Journal of Molecular Biology* 238: 658–668, 1994.