# On the Computational Power of DNA Annealing and Ligation*

May 25, 1995

Erik Winfree†

Computation and Neural Systems
California Institute of Technology
Pasadena, California 91125, USA
email: winfree@hope.caltech.edu

## Abstract

In [Winfree] it was shown that the DNA primitives of *Separate*, *Merge*, and *Amplify* were not sufficiently powerful to invert functions defined by circuits in linear time. Dan Boneh et al [Boneh] show that the addition of a ligation primitive, *Append*, provides the missing power. The question becomes, "How powerful is ligation? Are *Separate*, *Merge*, and *Amplify* necessary at all?" This paper proposes to informally explore the power of annealing and ligation for DNA computation. We conclude, in fact, that annealing and ligation alone are theoretically capable of universal computation.

## 1 Introduction

When Len Adleman introduced the paradigm of using DNA to solve combinatorial problems [Adleman], his computational scheme involved two distinct phases. To solve the directed Hamiltonian path problem, he first mixed together in a test tube a carefully designed set of DNA oligonucleotide "building blocks", which anneal to each other and are ligated to create long strands of DNA representing paths through the given graph. After this ligation phase, there ensue $n$ steps of affinity purification, whereby exactly the strands representing Hamiltonian paths are separated into a test tube ("the answer").

Richard Lipton [Lipton] subsequently refined the formalism for DNA-based computation. He did away with Adleman's first phase, ligation, and replaced it by starting all computations with a fixed set of DNA strands representing all $n$-bit strings. Lipton expanded on Adleman's second phase, separation, where he showed how all solutions to a given boolean formula $f$ can be separated into a test tube ("the answer"). The cost for the generality of this method is indicated by considering solving the Hamiltonian path problem: a straightforward method[1] takes about $n^3$ separation steps using Lipton's approach, compared to the $n$ steps used by Adleman.

*To appear in the Princeton DIMACS Technical Report on the April 4, 1995 workshop on DNA-based computers.

†This work is supported in part by National Institute for Mental Health (NIMH) Training Grant # 5 T32 MH 19138-05; also by General Motors' Technology Research Partnerships program.

[1] Let the graph have $n$ vertices and $e$ edges; $e \leq n^2$. The best boolean circuit I could devise uses $O(en \log n)$ gates to verify a Hamiltonian path. Another issue is that Adleman's ligation phase requires the synthesis of about $O(n + e)$ oligonucleotides, which is $O(n^2)$ if $e = O(n^2)$; whereas Lipton needs only about $4n \log n$ oligonucleotides to create his standard initial test tube of DNA. However, technology is becoming readily available for synthesizing many oligonucleotides in parallel very quickly (see *e.g.* [Chetverin]); the same cannot be said for the affinity purification steps, which will likely remain expensive. Comparing volume for a graph with $n/2$ edges out of each vertex, Adleman's method uses volume roughly proportional to $(\frac{n}{2})^n$, while Lipton's method uses a volume of $2^{n \log n}$, since it takes $n \log n$ input variable bits to specify a potential path.

We can conclude from this circumstantial evidence that much of the physical computational power Adleman was exploiting was in his first phase, where annealing and ligation were used. Lipton has explored the power of generalizing Adleman's second phase; we would like now to explore the power of generalizing Adleman's first phase.

An immediate stumbling block is that the chemistry of annealing is not fully understood. At best we can try to define some conditions under which the reactions are predictable, or at least under which it is reasonable to expect that the reactions could be made to be predictable.

## 2 Some Basic Annealing Reactions

The fundamental chemistry of DNA is based on the double helix and the principle of complementarity. Each strand of DNA is a covalently linked polymer, where each unit consists of a constant part (the sugar-phosphate "backbone") and one of either adenine, thymine, cytosine, or guanine (the bases A, T, C, G). Each strand is oriented; it has a 3' and a 5' end. When DNA forms a double-stranded helix, the strands must be anti-parallel, and complementary bases align (A with T, C with G); such strands are called Watson-Crick complementary sequences. DNA also takes on more complicated configurations, including triple helix, quad helix, super-coiled, and branched.

A surprising number of possibilities are available, some of which one may want, and many of which one may not want. DNA is a particularly easy molecule to work with, because it has evolved to be stable, typically unreactive, yet manipulable. RNA and protein, which have evolved to serve many enzymatic functions, are far more reactive, and thus it is less easy to predict how novel designs will behave in an experiment.

I will now comment on some reactions we may wish to exploit, presented in cartoon fashion (Figure 1). I will have to be more detailed with the reactions involved in the main thrust of this paper, where their computation-universality is demonstrated.

(A) This is the canonical annealing reaction for DNA. Two strands with complementary subsequences will form hydrogen bonds and hybridize at the matching base pairs. The rate constants for this reaction, which is reversible, depend on the temperature and salt concentrations, among other things. The melting temperature, above which the complex is not stable, depends upon the number of matching base pairs.

(B) A special case of the above, where the matched region occurs at the ends. Note that the two "sticky ends" (unmatched sequences) are available for further reactions with more DNA.

(C) The above reaction can be used to join two double-stranded DNA molecules with complementary sticky ends. If ligase is present in the solution, the nicks in the backbone of the product will be repaired by the formation of a covalent bond, resulting in two continuous strands.

(D) If mismatches occur flanked by matching regions, the unmatched DNA can bubble out.

(E) As above, except that the mismatch occurs here on both sides. Whether this structure is stable depends critically on the temperature and concentration of salts. For example, a rule of thumb is that the difference in melting temperature between a perfectly matched structure and an imperfectly matched structure is 1 degree per 1% mismatch [Wetmur].

(F) This is the simplest DNA branched junction. The assembly of these structures consists of course of sequential steps; only the end product is shown. This 3-armed junction is probably floppy. However, how floppy it is depends upon the exact sequence of base pairs in the oligonucleotides.

(G) This 4-armed junction is commonly known as a Holliday junction. The two horizontal strands tend not to be parallel, but skew. If the sequences along both strands are homologous, then a phenomenon called branch migration can occur, in which the crossover point drifts right or left.

(H) This is the most complicated structure we will consider. We will put it to good use later. It has been found to be fairly rigid and planar [Fu1]. Note the sticky ends. Other related double-crossover junctions are possible, depending upon the number of half-turns present in the helical regions. Ned Seeman calls this molecule "DAE" for double-crossover, antiparallel helical strands, even number of half-turns between crossovers. "DAO", with an odd number of half-turns between the crossovers, has an interesting topological difference: It consists of only 4 strands.

**Figure 1. Some basic types of annealing reaction.**

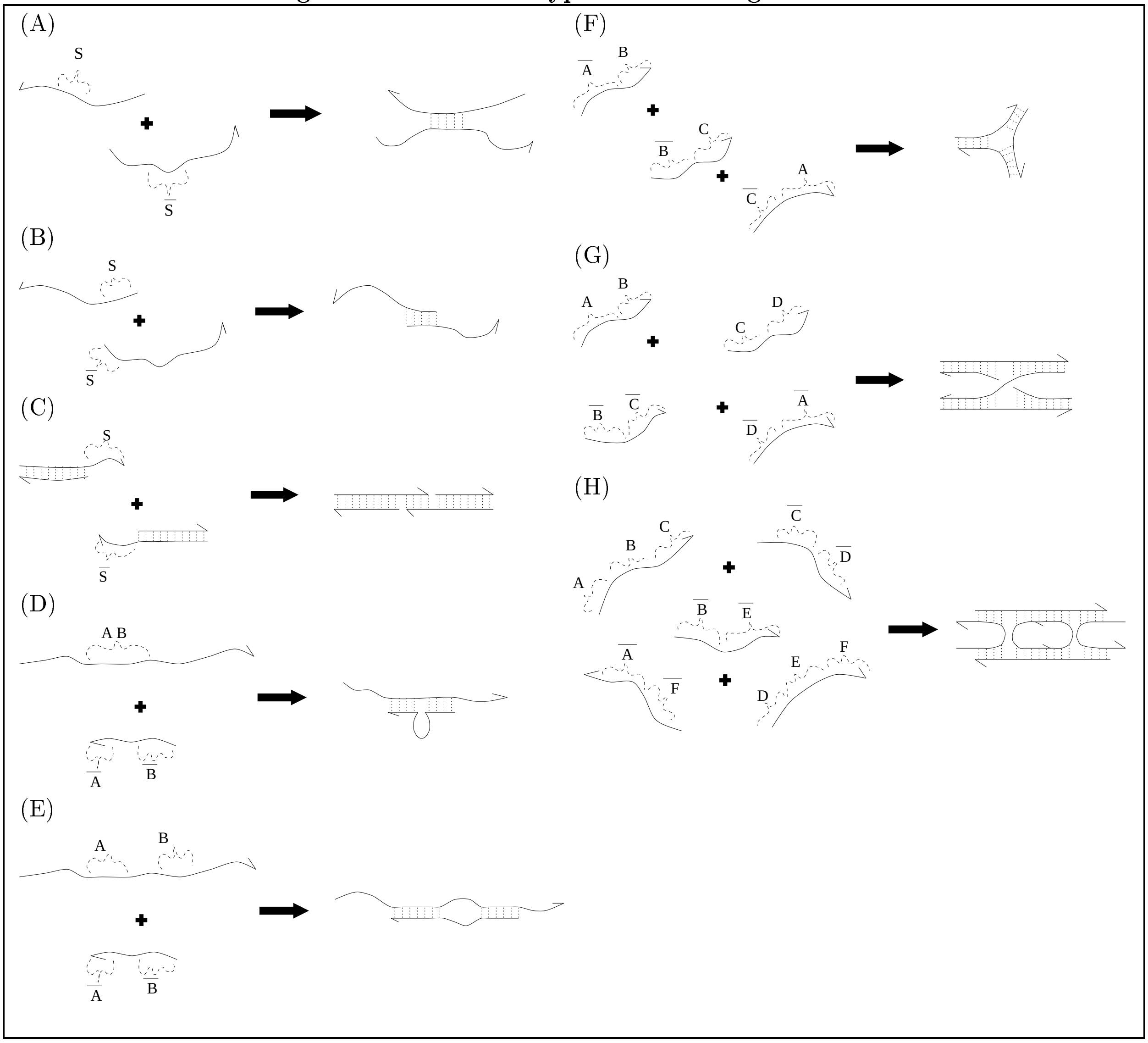


Figure 1: Curves represent single strands of DNA oligonucleotide. The half arrow-head represents the 3' end of the strand. Small lines between strands represent hydrogen bonds joining the strands. The helical structure of the DNA is not represented visually. Letters signify sequence motifs. A bar above a letter signifies the Watson-Crick complement of the motif.

All of the structures above have been made in the lab and their structures verified (see, for example, [Fu1]).

We would ultimately like a theory which could tell us, given a set of oligos, a temperature, and salt concentrations, what stable structures will form, as well as the kinetics. But this is a very complex task!

# 3 Operations using linear DNA

We will first briefly consider what computations can be performed using annealing and ligation of strictly linear DNA molecules. Many of the possibilities have already been discussed by other authors. For example, the techniques used by Adleman [Adleman] allow for the construction of all DNA representing strings accepted by a finite-state automata (also known as a regular language), using the annealing reactions (B) and (C) above. This is important, because it allows us to create a well-defined, somewhat interesting set of inputs on which to compute in parallel. Beaver has discussed how, in conjunction with polymerase, reactions (D) and (E) can be used to make copies of DNA with context-sensitive insertion, deletion, and replacement of substrings. In light of these powerful operations, it seems plausible that a "one-pot" linear DNA reaction could be designed which performs universal computation.

# 4 Operations using branched DNA

There are many possibilities for computation using branched DNA. However, since the general chemistry is not well understood, we will try to avoid ungrounded speculation by focusing on one concrete possibility. The rest of this section[2] will concentrate on how to assemble a large "weave" of branched DNA[3] which simulates the operation of a one-dimensional cellular automaton.

## 4.1 Background: Blocked Cellular Automata

This section develops a formal model of computation called blocked cellular automata[4] (BCA). We will later show how BCA can be simulated by DNA.

The operation of a BCA is diagrammed in Figure 2. As in the Turing Machine model, information is stored in an infinite one-dimensional tape, where each cell contains one of a finite set of symbols. The computation proceeds in steps, where in each step the entire tape is translated, according to a given rule table, into a new tape. The translation occurs locally and in parallel; pairs of two cells are read, and which two symbols are written is governed by look-up in a rule table[5]. It is of critical importance that the reading frame (which cells are paired together) strictly alternates from step to step.

The set of entries $\{(x, y) \to (u, v)\}$ is called the rule table, or the program, of the BCA. By appropriately designing the rule table, the BCA can be made to perform useful computation. In fact, BCA are computationally universal. A BCA with $k+3s$ symbols can simulate in linear time the operation of a Turing Machine with $k$ tape symbols and $s$ head states – the proof is analogous to that in [Lindgren]. Thus we can conclude that a BCA can be used to answer any question which can be phrased in terms of a computer program. Small BCA have been designed which sort lists of integers, compute primes, and many other tasks.

A few more comments are in order concerning the abstract model of blocked cellular automata. First we consider the finite-size case. In any attempted implementation of a BCA, we cannot actually construct an infinite tape. Thus boundary conditions become important. We consider the following cases:

(a) No update of boundaries. We start with a finite tape of length $2n$; at each step the tape become 2 cells shorter; and after $n$ steps the computation can proceed no further. This case is not universal.

(b) Inactive boundary conditions. Whenever there is an unpaired cell at either end of the tape, it is copied verbatim onto the new tape. The tape remains always the same size ($n$ cells), and thus there are only $k^n$ possible tapes. As the computation must begin to cycle after $k^n$ steps, this case is also not universal.

[2] The inspiration for this approach comes from the proof of the undecidability of the Tiling Problem (see [Grunbaum], Chapter 11).
[3] This is, clearly, highly speculative, but we hope not ungrounded.
[4] Blocked cellular automata are a 1D version of what Toffoli and Margolus call partitioning cellular automata in 2D [Toffoli].
[5] If the table contains multiple entries for a given pair of read symbols, then the BCA is said to be nondeterministic.

**Figure 2. Operation of a blocked cellular automaton.**

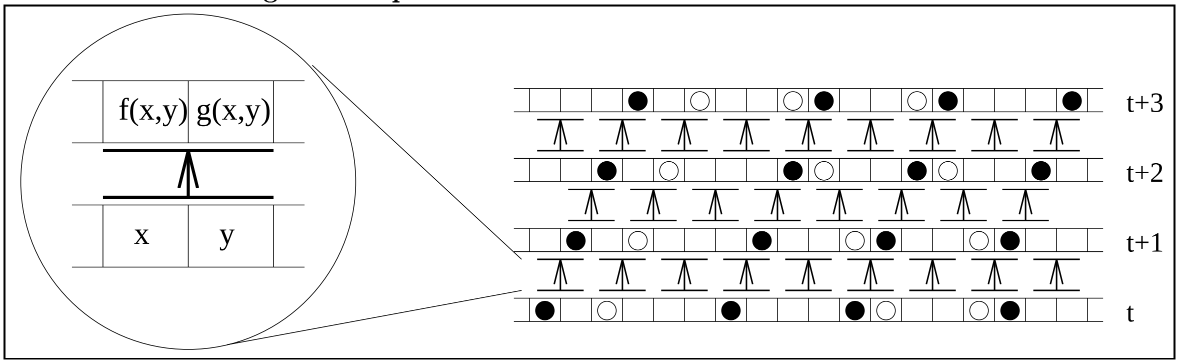


Figure 2: The tape of a BCA, divided into cells, is shown at the bottom right. Each cell contains one of three symbols: blank, black dot, or white dot. The tapes at successive time steps are stacked vertically above the initial tape. The inset, left, details the form of a rule table entry, which governs how new tapes are created.

(c) Periodic initial conditions. On either side of the input cells we specify a repeating pattern of symbols. Starting with just one copy of the periodic block on either side of the input, computation proceeds as in (a), but if the tape gets too short, we add another copy of the periodic block to either side of the input tape and start the computation anew[6]. This case *is* universal.

(d) Self-regulated boundary conditions. Depending upon what symbol is in the boundary cell, the new tape will either shrink (as in (a)) or expand by appending a new cell to the end of the tape. This case is also universal.

Finally, a word on how an answer is obtained from the BCA. This is a matter of convention. Typically, when the computation is done, the answer is written on the final tape. But how is it known when the computation is done? One possibility is that the tape stops changing; the system has reached a fixed-point. However in this paper we will consider that a computation is done when a special symbol, called the *halting symbol*, has been written for the first time anywhere on the tape.

## 4.2 Simulation of BCA by DNA

We will now show how to use DNA to construct a BCA. In this section we will optimistically show what chemical reactions we *hope* will occur; in the following section we consider potential difficulties in finding conditions such that they will in fact occur as we have described.

The DNA representation of the BCA tape is a little counter-intuitive, so we will explain by example. Figure 3 shows part of the DNA molecule encoding the initial tape (the input to the computation). To each tape symbol corresponds a short oligonucleotide sequence, which appears in the initial molecule as a sticky end overhang in the appropriate positions. The rest of the DNA in each segment does not vary with content, and is chosen to maximize structural stability. Note that the reading frame is implicit in the structural form of the DNA. Although Figure 3 is schematic, the 2D picture *is* meant to imply that the whole DNA complex is roughly planar. This is critical, and luckily, it is physically plausible.

There are a variety of ways to make the initial molecule. Note that the initial molecule can be thought of as consisting of several double crossover junctions (from Figure 1H, with the modification that the top and bottom strands are made to be an odd number of half-turns in length – see Figure 6 for detail) linked together by pieces of linear helical DNA. The sticky ends can be designed such that only this unique molecule will self-assemble[7]. Ligase can be added to make the segments of the initial molecule covalently bonded.

[6]By memorizing boundary cells, we can avoid re-computing any cells and make the computation more efficient.

[7]It is easy to see that sticky end sequences can be chosen, using the same techniques as Adleman (see Section 3) , such that a periodic initial molecule will form, creating periodic initial conditions as mentioned in section 4.1 (c) above. Similarly, a regular language of inputs could be made in parallel.

**Figure 3. Encoding the initial tape in a DNA molecule.**

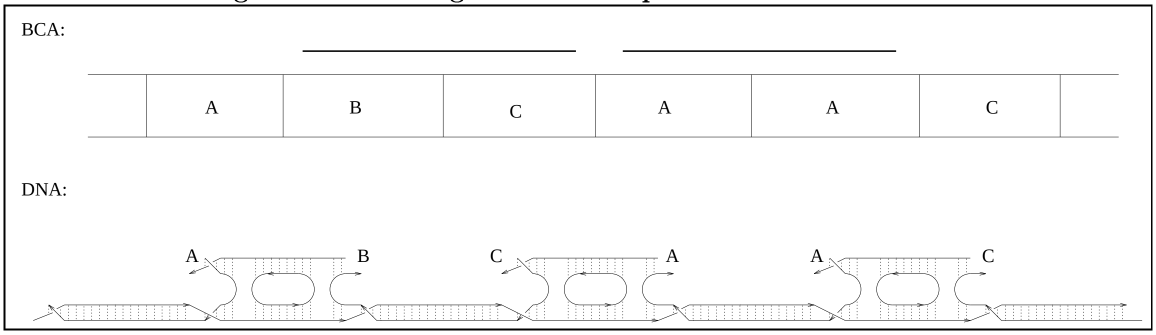


Figure 3: The sequence of sticky ends in the initial molecule encodes the initial tape of the BCA. Thus 'A' denotes a symbol in the BCA diagram, whereas in the DNA diagram it denotes the unique sequence of bases associated with that symbol.

We will now explain how the program, that is the rule table, of the BCA is represented in DNA. For each rule, *e.g.* $(x, y) \rightarrow (u, v)$, we create a double crossover molecule whose sticky ends on one helix are $\overline{x}$ and $\overline{y}$, and on the other helix $u$ and $v$[8] (see Figure 6). All such rule molecules are added to the solution containing the initial molecule. As shown in Figure 4, what is required for computation is that rule molecules will anneal into position if and only if both sticky ends match.

**Figure 4. Rule table molecules assemble into the lattice.**

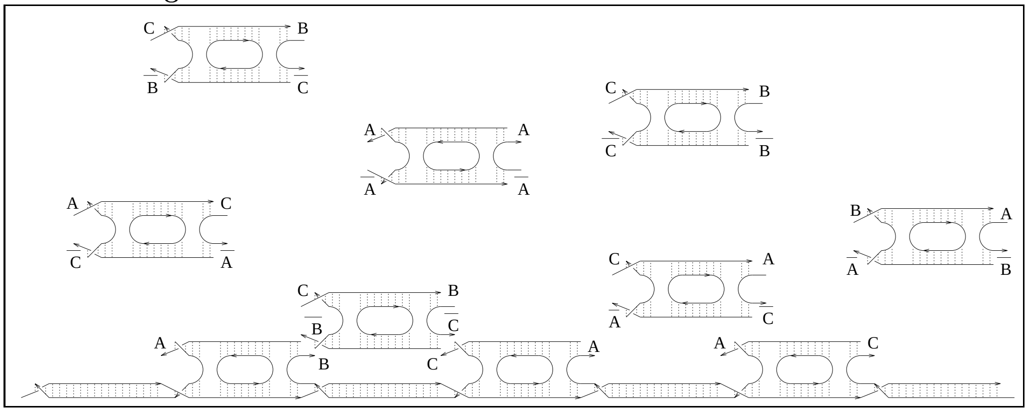

Figure 4: We see free-floating rule table molecules above and the initial molecule at the bottom (both correspond to the BCA in Figure 2). A rule table molecule, with sticky ends $\overline{B}$ and $\overline{C}$, is about to anneal to the initial molecule. At the left, a rule molecule which matches only at $\overline{A}$ will ultimately not stick. Note that the rule molecule with sticky ends $\overline{A}$ and $\overline{A}$ will also not stick, because the orientation of its strands is wrong; this rule molecule will be useful on alternate levels of the lattice.

Eventually, a triangular lattice of linked DNA will form, simulating a triangular region of a BCA corresponding to boundary conditions (a) or (c) in Section 4.1 above (see Figure 5). Boundary conditions (b) and (d) can be simulated by using special rule molecules for the edge of the lattice; the details are not presented here. Note that each level of the lattice has a single strand of DNA which travels the entire length of the lattice at that level, and where the coded symbols occur in the sequence in in which they occur in the BCA at time $t$.

Finally we ask, how can we access the output of the computation? This breaks down into two questions: How do we know *when* the computation is done? And *what* is on the tape at that point? There are many

[8]The lengths of all parts of the rule molecules are chosen to be constant for simplicity, but it is conceivable that by using variable length as well as sequence to encode symbols, greater specificity could be achieved.

**Figure 5. The DNA lattice resulting from a finite initial molecule.**

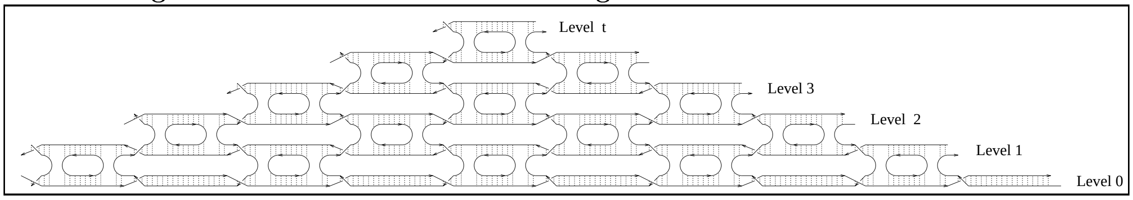


Figure 5: At the chosen annealing temperature, which is above the melting temperature for $s$ base-pair annealing but below the melting temperature for $2s$ base-pair annealing, no more rule molecules can stably attach to this structure. However, if the bottom level (the initial molecule) were extended, then a larger triangle could form. $s$ is the length of the sticky ends in the rule molecules.

possible approaches to take; here we will merely sketch one. As mentioned above, we will consider the computation to be done when a special halting symbol is written on the tape[9]. In DNA, this corresponds to the special sticky end motif being incorporated into the lattice. When this occurs, the motif will be present as a double-stranded molecule for the first time, and this site can be be chosen as the recognition domain for a binding protein[10], which could, for example, subsequently catalyze a phosphorescent reaction, turning the solution blue. To determine what is "on the tape" at this point, it is necessary to extract the single strand of DNA corresponding to the final level of the BCA. To do this, first add ligase to covalently bond all the annealed segments[11]. Then add resolvase to break all the crossover junctions[12]. Finally, heat to separate the strands, and use affinity purification to extract the strand containing the halting motif. Amplify and sequence that strand however you desire (*e.g.* via PCR and standard sequencing gels).

To summarize the model suggested here, a computation would proceed as follows.

1. First, express your problem via computer program. Convert that program into a (possibly nondeterministic) blocked cellular automaton.
2. Create small molecules (H-shaped and linear) which self-assemble to create the initial molecule (or initial molecules, if search over a FSA-generated set of strings is desired). Add ligase to strengthen the molecule.
3. Create small H-shaped molecules encoding the rule table for your program.
4. Mix the molecules created in steps 2 and 3 together in a test tube, and keep under precise conditions (temperature, salt concentrations) as the DNA lattice crystallizes.
5. When the solution turns blue, ligate, cut the crossovers, and extract the strand with the halting symbol.
6. Sequence the answer.

[9] At this point other parts of the tape will typically "not know" that the computation is done, so the lattice will continue to grow. However, it is also possible to design the cellular automaton such that all cells go into a special state to halt computation at the same time (the Firing Squad Problem, see *e.g.* [Yunes]), thereby allowing us to design linear pieces of DNA which fit into the gaps at the final level of the lattice, so that it cannot grow further. This may make extraction of the final tape configuration easier.

[10] The protein must have an active bound form, and inactive unbound form. Furthermore, we must be sure it doesn't bind to rule molecules in the solution.

[11] It is a valid concern that ligase may not be able to bind to any but the outermost strands in a lattice. It may be better to reverse the order of the ligase and resolvase steps.

[12] Although a resolvase has been shown to cut crossovers in double-crossover molecules [Fu2], it is unknown whether the enzyme will be functional on the inner strands in the lattice. However, the enzyme may be able to, at diminished speed, work from the edges in.

## 4.3 Analysis and Estimates. Will it work?

Let's begin the analysis optimistically. The above construction is just one implementation possible in a general class that might be called "crystal computation"[13]. In this class, we design a system where we can tailor-make the energy (and hence free energy) as a function of the configuration. We design it such that the lowest energy state (or in our case, the lowest free-energy state at a given temperature) uniquely represents the answer to our computation. This is closely related to the approach taken by J. J. Hopfield [Hopfield] in his seminal work on neural networks. In our case the lowest energy configuration is one where every rule molecule has all four sticky ends bound. Given the presence of the initial molecule, this can only occur if the computation proceeds as desired.

The above analysis is a simplification that fails to take into consideration many aspects of the proposed implementation. For example, it completely ignores the dynamics involved; one simply anneals at a slow enough schedule, the argument goes, and the crystal is the result. Whereas in fact the crystallization proceeds at the edges only, according to kinetics that significantly influence the result.

Can a temperature be found such that two sticky ends bound is stable, while one sticky end bound is unstable? In other words, let $T_0$, $T_1$, and $T_2$ be the melting temperatures for a rule molecule fitting into a lattice slot where respectively 0, 1, and 2 of the sticky end pairings match. We want to keep the test tube at a temperature $T$ such that $T_0 < T_1 < T < T_2$. This should be possible, but how large is the difference between $T_1$ and $T_2$? Although this is unknown for the particular molecules we use, we can get some idea by looking at what's known about linear DNA annealing. For example, under standard conditions 20 base-pair oligonucleotides (representing rule molecules with two length 10 sticky ends bound) melt at 70° C, while 14 base-pair oligonucleotides (representing rule molecules with only one length 10 sticky end bound, and the other matching partially) melt at 58° C [Wetmur]. $T = 65°$ C would then discriminate the two cases. However, the analogy of rule molecules with two separate binding domains to variable-length oligonucleotides with continuous binding domains is questionable.

A definitive answer to "But will it work?" requires a chemist's knowledge and actual experiments. But we can immediately bring some more concerns to light. Since I do not have answers to them, I will merely mention them in passing. First, to read out an answer of more than one bit, our implementation requires ligating the rule molecules and cutting them with resolvase. It is not at all clear that, in the crowded confines of the DNA lattice, either ligase or resolvase will have room enough to perform its job[14]. Second, it is possible that, at a low rate, incorrect rules will be incorporated into the lattice. If this occurs, the computation is ruined. It is thus not clear at this time what yields of correct computation are to be expected, and whether a means could be devised to separate the good from the bad. It is additionally conceivable that stable structures form in the solution unconnected to the initial molecule. For example, four rules molecules could connect in a stable "diamond"; we might think that these complexes will only rarely be formed, because the intermediate steps are unstable (only one sticky end joins molecules), and for similar reasons they would grow slowly. However, they and other types of spurious connections and tangles could form, ruining the computation. A final concern is that there may be some systematic molecular stress or strain that comes into play when building a large crystal, and that beyond a certain size tearing would result. All these issues, and surely others, deserve more attention and study.

If for the moment we suppose that the implementation operates correctly, let us consider what advantage would be derived. Take the following with a bucket of salt: First, a small rule molecule (see Figure 6 for a close-up) consists of 50 base-pairs of DNA, sufficient for sticky ends of length 5, which gives us $\approx 10$ symbols[15]. That's 33 K Dalton / rule molecule, with a size probably less than 20 x 44 x 85 Angstroms, for 3 bits / rule molecule.

[13] It has been suggested that we shouldn't use the term "crystal", because it has a well-defined special meaning. At best, our constructions yield "pseudo-crystals", because any useful computation is aperiodic. We beg the reader to give us slack in using this term.

[14] If there is an angle between the plane of the lattice and a rule molecule which has just fit in place, then in our construction, an opposite angle is formed when a rule molecule fits into the subsequent layer. Consequently, the 2D lattice, rather than being perfectly planar, folds back and forth like a paper fan, which we call a "corrugated" lattice. The corrugated lattice exposes more of the double helix strands in each rule molecule, possibly making the strands more accessible to ligase but making the

**Figure 6. Detail of a small rule molecule.**

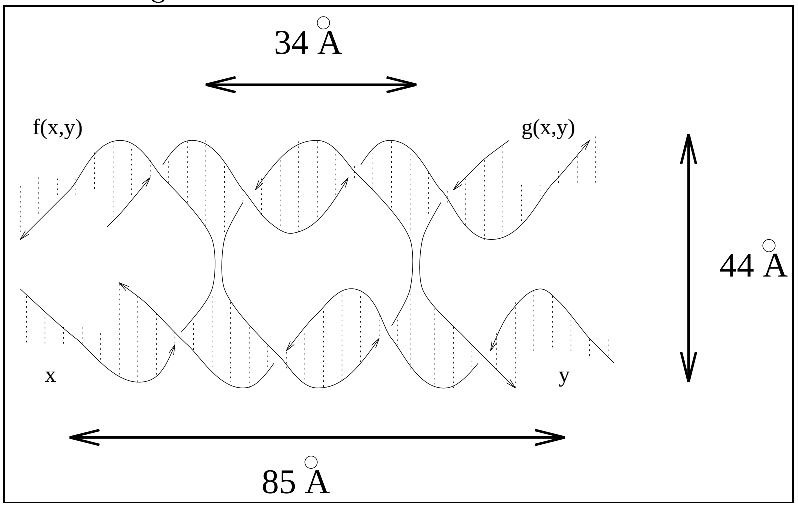


Figure 6: This is the smallest DAE/even style rule molecule possible. It has sticky ends of length 5, and internal region of length 10. Every base pair is shown.

Assessing speed is even more speculative. Suppose we perform a computation of a 10000-cell BCA with inactive boundary conditions, and compute for 10000 time steps. Suppose it takes 1 second for a rule to fit in when its slot is exposed. Since the 5000 slots are simultaneously exposed, all should be filled in approximately 1 second on average. This leads to a rough estimate of 3 hours for computing the $10000^2$ cell lattice. Using 1kg of DNA, we could assemble $10^{19}$ rule molecules, that is, $10^{11}$ such calculations in parallel. That leads to a total of $10^{15}$ operations per second[16]. There is no lab work to be done during this the major stage in the computation. Of course time would also be required in the input and output stages.

## 4.4 Open questions, extensions, and other speculation.

In addition to the essential question of whether the ideas above can be made to work in the lab, there are many other issues to be investigated.

**How energy-efficient is crystal computation?** It is interesting to note that what might be called the computation proper (crystallizing the DNA lattice) theoretically requires no energy at all; in fact, crystallization must be exothermic. Of course, a great deal of energy may be used to heat the mixture up, or to pulse the temperature to dissolve defects. Furthermore, the input and output stages require synthesis and analysis of DNA molecules, and thus also much energy. Our proposal is possibly the most nearly implementable example of the principle that computation is free, but input and ouput are costly [Bennett].

**Why use the DAE structure for rule molecules?** Clearly the particular choice of molecule is not of intrinsic importance to the idea of this construction. The logical essence is to have an "H"-shaped molecule with four designable sticky ends. At its simplest, one could imagine making the "H" out of two chemically cross-linked strands of DNA (Figure 7a). Another alternative is the slightly larger single crossover Holliday junction. However, it is important for the construction of the lattice that the two linear pieces in the "H" be planar; Holliday junctions have been shown to prefer a (flexible) 60° skew angle [Eis]. The chemically linked strands imagined above have not yet been characterized. The reason we propose the large double crossover

crossovers less accessible to resolvase.

[15] We optimistically require only 2 mismatches between sequences representing differing symbols. We also require the complement of a symbol's sequence does not code for a symbol, and that every code sequence has 3 C-G bonds and 2 A-T bonds, for more consistent melting temperatures.

[16] This compares to 300 GFLOPS ($\approx 10^{14}$ basic operations per second) attainable by the best modern supercomputers, *e.g.* a 7000 processor Intel Paragon. Of course, the "operations" we compare are apples and oranges.

**Figure 7. Alternative Topologies for 2D Lattice.**

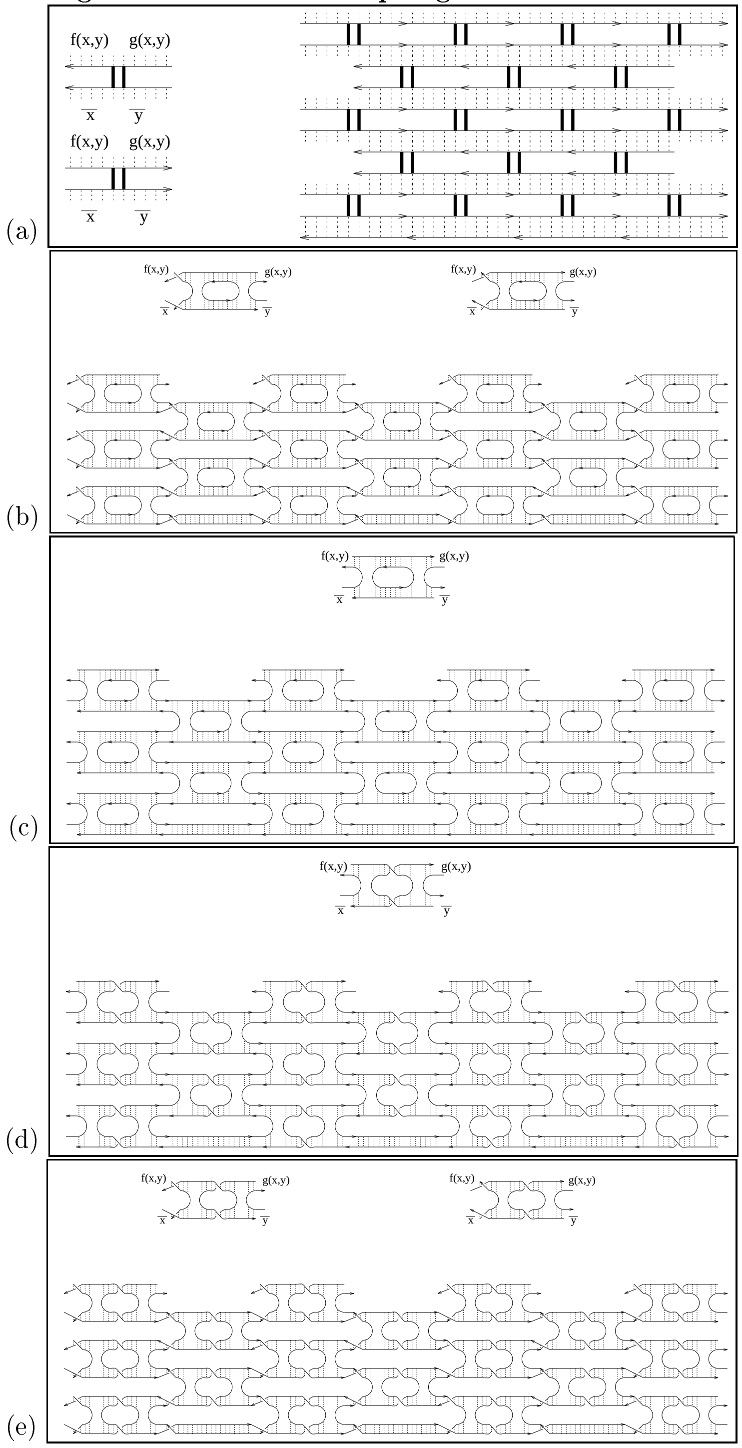


Figure 7: (a) Rule molecules based on cross-linked DNA. (b) DAE rule molecules with odd-length spacing. (c) DAE rule molecules with even-length spacing. (d) DAO rule molecules with odd-length spacing. (e) DAO rule molecules with even-length spacing.

molecules[17] is that they have already been characterized in the lab and are thought to be rigid (which may help prevent tangled lattices) and planar [Fu1]. We chose DAE in preference to other topological variants of double crossover molecules, such as DAO, because the topology of the rule molecule leads to a different "weave" of DNA strands in the lattice (Figure 7bcde). We prefer to have a single strand which, if covalently linked, runs along an entire level of the lattice, thus encoding the BCA state for that time step.

**Why keep around the entire history of the computation?** Only the most recent level is necessary for the next step of the computation. *Open question*[18]*: Can condition be found such that the bottom of the lattice is dissolving while the top of the lattice is growing?* Rule molecules which dissolve at the (hotter?) bottom of the lattice could later be re-used at the (colder?) top.

**Automatic programming by evolving rule molecules.** Suppose we are interested in finding a small BCA program which generates a particular string, or set of strings. Speculatively, we might begin with a nondeterministic set of all possible rule molecules of a particular size, including some molecules for non-deterministically constructing initial molecules. We grow some $10^{18}$ lattices, and somehow extract those which compute the desired string. The rule molecules present in these lattices are known to be sufficient to compute the string, but they probably do not contain all possible rules. We now dissolve the "good" lattices and somehow amplify the rule molecules present. Letting lattices grow again, and selecting again for the desired string, we further reduce the nondeterminism of the rule molecules present. We can also consider adding a tiny amount of ligase, thus occasionally creating larger rule molecules from smaller ones – a form of "compiling". Perhaps after a few iterations we look and see what rule molecules are present, or – presuming there is still some nondeterminism – look at what other strings they form. This process is closely related to universal search and can be used, for example, for Kolmogorov complexity based induction[Solomonoff].

**Why a 1D BCA? Why not build a 3D lattice to simulate a 2D BCA?** We started with 1D BCA because they can be immediately explored used existing DNA technology. Two dimensions offers several advantages, however, such easier design of efficient computations. Perhaps more importantly, in higher dimensions it becomes easier to design error-tolerant rules [Gacs]; intuitively, point defects in 2D can be filled-in from adjacent correctly-computed cells, while in 1D a point defect severs communication between the left and right side. *Open question: Can the DNA rule molecules be modified so as to build 3D DNA lattices?* Speculatively, one could propose a variant of the double crossover Holliday junction, the "multiple strand double crossover junction" (Figure 8), as a means to implement the read-4, write-4 operation required by 2D blocked cellular automata (see e.g [Toffoli], Ch. 12). Unfortunately, the proposed building-block molecule has not yet been synthesized.

**Figure 8. A possible 3D lattice of DNA for simulating 2D BCA.**

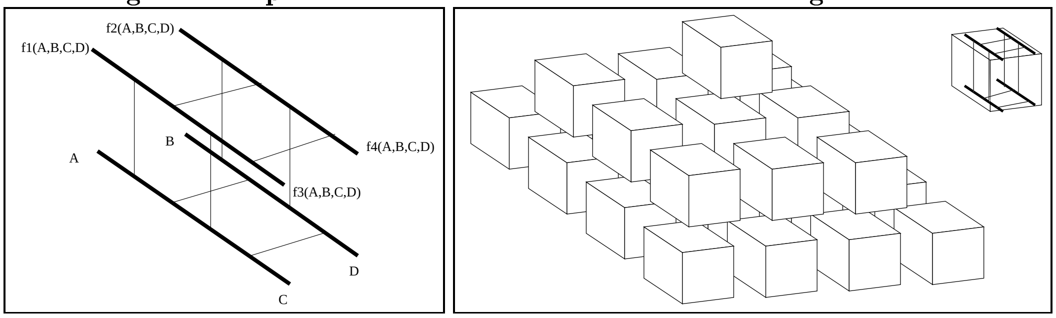


Figure 8: Four DNA double helices may be bound together by crossover junctions (left). Sticky ends determine 2D BCA rules as the rule molecules assemble in an alternative cubic lattice (right).

**Potential uses in nano-technology.** This paper has suggested an approach to molecular computation via programmable self-assembly. Programmable self-assembly may have other applications. *Open question: Can cellular automata generated lattices be used to define ultra-high resolution electronic circuits?* One

[17] Ned Seeman suggested we consider double crossover molecules as an improvement over the more awkward branched junction constructions we were originally considering.

[18] Suggested by Len Adleman, private communication.

possibility, along the lines investigated by Robinson and Seeman [Robinson], would be to conjugate nanowire onto individual rule molecules, such that when the rule molecules fit together, an electrical circuit is formed. This proposal differs from Robinson and Seeman's suggestion in that whereas they envisioned a periodic lattice of identical memory cells, we suggest that cellular automata rules could be used to build more complicated circuits, either in 2D or 3D.

**Why use DNA at all?** The principle of computing via crystallization is not restricted to DNA. *Open question*[19]*: Can non-DNA-based molecules could be used to design desired computations carried out on the surface of a growing crystal?*

# 5 Comparison with other approaches

Perhaps the most practical suggestion for universal computation via DNA is that of Boneh, Dunworth, Lipton, and Sgall [Boneh]. Their approach makes straightforward use of well understood laboratory techniques for manipulating DNA. They are able to simulate nondeterministic boolean circuits, which seems very efficient for some calculations, and which gives them universal computational ability. Because circuits allow non-local interactions of variable, circuits can be very compact. However, it should be pointed out that the computation requires a lab technician to sequence operations on multiple test tubes; the logic of the program being computed is external to the DNA, which is used as a memory. Small scale computations could be immediately attempted with reasonable chance for success; however due to the weakness of single-stranded DNA and other factors, it is not clear how this approach will scale.

Other authors have proposed DNA implementations of Turing Machines directly (*e.g.* [Beaver], [Smith], [Rothemund]). The approaches vary from using PCR to relying on restriction enzymes. These approaches show promise, although the reliability and efficiency of the steps is unclear. Furthermore, single-tape, single-head Turing Machines are particularly cumbersome logically; circuits will typically compute the same function in many fewer steps (and single steps take comparable time in both systems – on the order of hours!). In short, although they are of theoretical interest, it is unlikely that anyone will actually go into the lab and solve problems this way.

Our hypothetical cellular automaton implementation differs in a number of ways: First and foremost, our proposal is a "one-pot" reaction. Dump in the rule molecules encoding your problem, and all the logic of the computation is carried out autonomously. No lab work is involved. Furthermore, in addition to running a massive number of computations in parallel, each cellular automaton performs its own computation in parallel – thus fully exploiting the parallelism available. The major and significant drawback of our proposal is that it makes use of chemistry which is not yet fully understood, and thus going into the lab to do a computation this way would be a real technical challenge.

The main conclusion of this paper is that annealing and ligation *alone* may be sufficient for universal "one-pot" DNA computation. Whether the particular scheme envisioned here can be made to work in the lab is a matter for further research. In any case, it is clear that better experimental characterization of the chemistry of annealing is required, and may open up new possibilities for DNA based computation.


# 6 Acknowledgments

I would like to thank Paul W. K. Rothemund and Sam Roweis for their stimulating discussion. I am indebted to Ned Seeman for many excellent suggestions, as well as fundamental research on the biochemistry this proposal hopes to exploit; and to Len Adleman for inspiration and great discussions. John Baldeschwieler, Tom Theriault, Marc Unger, Sanjoy Mahajan, Carlos Brody, Dave Kewley, Pam Reinagel, Al Barr, and Stuart Kauffman gave many useful suggestions. Thanks to my advisor John Hopfield for his support and encouragement.


[19]Suggested by Stuart Kauffman, private communication.

## References


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

[Beaver] Beaver, Don. A Universal Molecular Computer. Penn State University Tech Report CSE-95-001

[Bennett] Bennett, Charles H. The Thermodynamics of Computation – a Review. *International Journal of Theoretical Physics* 21 (12) : 905–940, 1982.

[Boneh] Boneh, Dan, C. Dunworth, R. Lipton, and J. Sgall. On Computational Power of DNA. To appear.

[Chetverin] Chetverin, Alexander B., and Fred Russell Kramer. Oligonucleotide Arrays: New Concepts and Possibilities. *Bio / Technology* 12:1093–1099, November 1994.

[Eis] Eis, P. S., and D. P. Millar. Conformational Distributions of a Four-Way Junction Revealed by Time-Resolved Flourescence Resonance Energy Transfer. *Biochemistry* 32 (50) : 13852–13860, 1993.

[Fu1] Fu, Tsu-Ju, and Nadrian C. Seeman. DNA Double-Crossover Molecules. *Biochemistry* 32: 3211–3220, 1993.

[Fu2] Fu, Tsu-Ju, Börries Kemper, and Nadrian C. Seeman. Cleavage of Double-Crossover Molecules by T4 Endonuclease VII. *Biochemistry* 33: 3896–3905, 1994.

[Gacs] Gacs, P., and J. Reif. A Simple Three-Dimensional Real-Time Reliable Cellular Array. *Journal of Computer and System Sciences* 36 (2) : 125–147, 1988.

[Grunbaum] Grünbaum, Branko, and G. C. Shephard. *Tilings and Patterns.* W. H. Freeman and Company, New York, 1987.

[Hopfield] Hopfield, J. J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities.

[Lindgren] Lindgren, K., and M. Nordahl. Universal Computation in Simple One-Dimensional Cellular Automata. *Complex Systems* 4 (3): 299–318, 1990.

[Lipton] Lipton, Richard. Using DNA to Solve NP-complete Problems. *Science* 268: 542–545 (Apr. 28) 1995.

[Robinson] Robinson, Bruce H., and Nadrian C. Seeman. The Design of a Biochip: A Self-assembling Molecular-scale Memory Device. *Protein Engineering* 1 (4) : 295–300, 1987.

[Rothemund] Rothemund, P. W. K. A DNA and Restriction Enzyme Implementation of Turing Machines. This volume.

[Smith] Smith, Warren D., and Allan Schweitzer. DNA Computers in Vitro and Vivo. NECI Technical Report, March 20, 1995.

[Solomonoff] Solomonoff, Ray. The Applications of Algorithmic Probability to Problems in Artificial Intelligence. In *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors). Elsevier Science Publishers B. V., North Holland, 1986.

[Toffoli] Toffoli, Tommaso, and Norman Margolus. *Cellular Automata Machines.* MIT Press, Cambridge, MA, 1987.

[Wetmur] Wetmur, James G. DNA Probes: Applications of the Principles of Nucleic Acid Hybridization. *Critical Reviews in Biochemistry and Molecular Biology* 36 (3/4) : 227–259, 1991.

[Winfree] Winfree, Erik. Complexity of Restricted and Unrestricted Models of Molecular Computation. This volume.

[Yunes] Yunes, J. Seven-state Solutions to the Firing Squad Synchronization Problem. *Theoretical Computer Science* 127 (2) : 313-332, 1994.