A DNA and restriction enzyme implementation of Turing Machines.

(Turing machines; Universal Turing machines; recombinant DNA; nonpalindromic endonucleases; class IIS nucleases; biological computation; molecular computation; chemical computation, DNA computation)

Paul Wilhelm Karl Rothemund

533 South Hudson Apt. #1, Pasadena CA 91101
Tel. (818) 584-1807, Email: pwkr@alumni.caltech.edu.
WWW: http://www.ugcs.caltech.edu/~pwkr/oett.html

ABSTRACT

Bacteria employ restriction enzymes to cut or *restrict* DNA at or near specific words in a unique way. Many restriction enzymes cut the two strands of double-stranded DNA at different positions leaving overhangs of single-stranded DNA. Two pieces of DNA may be rejoined or *ligated* if their terminal overhangs are complementary. Using these operations fragments of DNA, or oligonucleotides, may be inserted and deleted from a circular piece of plasmid DNA. We propose an encoding for the transition table of a Turing machine in DNA oligonucleotides and a corresponding series of restrictions and ligations of those oligonucleotides that, when performed on circular DNA encoding an instantaneous description of a Turing machine, simulate the operation of the Turing machine encoded in those oligonucleotides. DNA based Turing machines have been proposed by Charles Bennett but they invoke imaginary enzymes to perform the state–symbol transitions. Our approach differs in that every operation can be performed using commercially available restriction enzymes and ligases.

INTRODUCTION

The concept of a molecular computer, whose basic operations would be performed chemically instead of electronically, has long intrigued us with the possibility of storing and manipulating information at densities impossible to realize with current computers. For many years, the search for a chemistry rich enough to make a molecular computer has been a focus of our efforts to build them. To many, the chemistry of life with its myriad of enzymes and informationally rich nucleic acids seemed a good candidate—and it has provided us with our first success. Last year, Leonard Adleman used the chemistry of DNA to solve the directed Hamiltonian path problem [**Ad1**]. This was a very important first step towards realizing molecular computation—similar combinatorial approaches may prove to be the best way to solve NP complete problems. Such an approach does not, however, give us a *programmable* molecular computer that can solve any problem for which we can write a algorithm.

In order to construct a general molecular computer some Universal model of computation (*e.g.* a digital computer, neural network, Turing machine, *etc.*) must be expressed in chemistry. Numerous workers have proposed such translations using theoretical chemistries. Charles Bennett first likened the operation of RNA polymerase to a Turing machine in 1973 [**Ben1**]. In 1982 [**Ben2**] he gave a schematic description of a DNA Turing machine using imaginary enzymes capable of recognizing and changing single bases of DNA. Others have since proposed the construction of chemical computers using different sets of hypothetical chemical species and different models of computation. Hjelmfelt *et al.* [**Hj**] describe the construction of chemical neural networks which in turn, they proposed, might be used to make other general computers like Turing machines. Models like these that detail how a computation would proceed if we had certain chemicals at our disposal probably must await breakthroughs in protein engineering to be implemented.

Extensions of Adleman's approach ([**Ad2**], [**Bon**]) *can* implement universal models of computation. Adleman [**Ad2**] presents a series of operation on "DNA test tubes" that simulate a memory model of computation. Boneh *et al.* give a method for simulating nondeterministic boolean circuits. In both of these schemes each reaction performed corresponds to a statement in a computer program or an element of a circuit. This means that the "program" for these computers is actually executed by the chemist.[1] Universal programs can be written for these models but it is suspected that they would be large and require a large number of distinct steps to be implemented.

In this paper we present a method for encoding the transition table of a Turing machine with oligonucleotides, representing a Turing tape, head position, and state, as a single molecule of DNA, and effecting transitions using restriction enzyme chemistry. A total of 6 distinct chemical steps are repeated to simulate a given Turing machine. All of the reagents used in this encoding are commercially available (*New England Biolabs*) and all of the chemical operations are routinely performed by molecular biologists. Using this method a Universal Turing machine (capable of simulating any Turing machine and hence any algorithm) can be constructed.

This paper includes: a review of the Turing machine formalism; a review of the structure of DNA; the operation of restriction enzymes on DNA; a schematic encoding of the three state Busy Beaver Turing machine; the translation of this schematic into real restriction enzymes and oligonucleotides; a description of its extension to a Univeral Turing machine; and a discussion on the practicality of this approach.

1 TURING MACHINES

1.1 Models of computation

A Turing machine [**Tu**] is a *model of computation*—a way of representing and performing a given computation. Turing machines are mathematically equivalent to many other models of computation—cellular automata [**Lind**], neural networks [**Si**], and digital computers [**Ho**]. Because none of these models of computation (or any other we have found) is more powerful than the Turing machine model we believe that Turing machines embody what we mean when we say something is *computable* (Church's Thesis.) That is, anything for which we can write a procedure, or an *algorithm*, can be computed by a Turing machine. Because of its simplicity and equivalence to other models of computation, Alan Turing's model has allowed us to prove many important results about the nature and limits of computation (*e.g.* the undecidability of the halting problem, and the existence of uncomputable functions.) A correspondence between a DNA computer and Turing's model puts DNA computation on equal footing with *any* other model of computation and allows it to implement *any* algorithm.

1.2 An informal description of Turing machines

A schematic representation of a Turing machine (TM) operating on its tape is given in Figure 1. The tape can be thought of as a sequence of memory *cells* extending indefinitely[2] in both directions. Each cell can store a single symbol from the set $S = \{s_0, s_1, \ldots, s_N\}$. The Turing machine has two parts: a *head*, and a *finite control.* The head points to one cell of the tape and may read a symbol from that cell, write a symbol to that cell, or move right or left to an adjacent cell. At each timestep in the operation of the Turing machine the head performs a compound operation composed of a single read, write, and move. The finite control—the "brain" of the Turing machine—directs the head. A special cell in the finite control contains the *state* of the machine, a member of the set $Q = \{q_0, \ldots, q_P\}$. The rest of the finite control houses the *transition table* which defines how the finite control will instruct the head when the head points to a given symbol s_v and the finite control is in a given state q_x.

For every possible pair (s_v, q_x) the transition table gives a triple (s_w, q_y, m) where s_w is the symbol written by the head, q_y is the new state for the machine, and m is the movement made by the head—L, R, or H, for left, right, and halt, respectively. If m = H the machine enters the a special state q_0, known as the *halting* state, and the computation stops. This allows us to leave q_0 out of the specification of a Turing

[1] It's easy to get a feel for how these models work by simulating them at home: just write out the contents of your computer's memory on little slips of paper and push them around according to a program. The little slips of paper *do* perform a computation, but they hardly evoke the image of a computer that we are familiar with—one of programming a computer, typing **run**, and watching it go. Nevertheless, DNA memory models have a real advantage over paper pushing computers and, perhaps, electronic ones—they allow us to push around 10^{14} little pieces of DNA paper at once! (In [**Ad1**] 4 x 10^{14} DNA molecules are used to represent edges in a graph.)

[2] *Indefinitely*, in this context, means that if ever our Turing machine runs out of tape we append a few more cells.

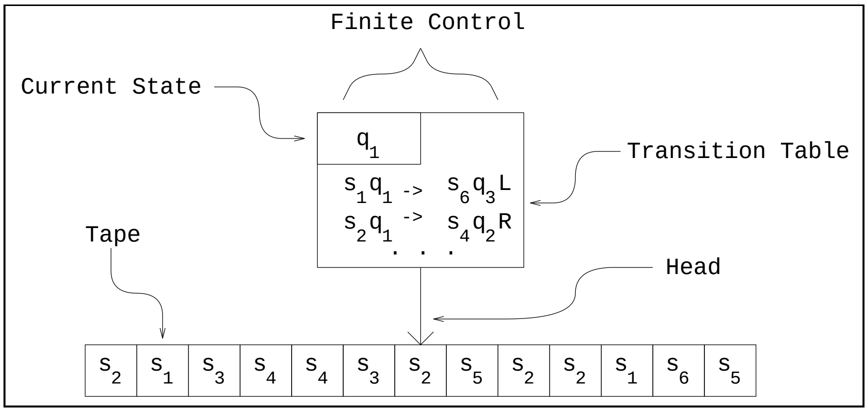

Figure 1: Turing machine model of computation.

machine and consider only the non-halting states. Indeed when we state the *size* of a TM we call a machine with *j* symbols and *k* *non-halting* states a *j* x *k* TM.

Instead of dividing the Turing model of computation into "hardware" and "magnetic media" as we do when we group the head, the transition table, and the state of the finite control together as the Turing machine and place the tape by itself, we might group these elements of the model along different lines: constant elements and variable ones. The variable elements of the model, the head position, the state of the finite control, and the contents of the tape are, taken together, known as an *instantaneous description* or *ID* of a Turing machine. The remaining part of the model, the transition table, is constant and we call it the *machine* part of the model. The Turing machine model, divided in this way, allows us to picture the machine as an *operator* that acts on an ID at time T and produces a new ID at time T+1 [**Ab**]. This is the image of a Turing machine that guides the DNA implementation of Turing machines given in this paper.

1.3 An example: Simulation of a 3 state Busy Beaver machine

The Turing machine we show here is a solution to the well known Busy Beaver problem for three state Turing machines.[3] This Busy Beaver machine (BB-3) has a two symbol alphabet S = {B, W} and three (non-halting) states, Q = {q_1,q_2,q_3}. The transition table for the machine is given in cartoon form in Figure 2. The symbols B and W are represented by black and white boxes. Smaller boxes, representing the states q_1, q_2, and q_3 have been assigned three shades of gray. A movement to the left is given by a left arrow and a movement to the right by a right arrow. The next move for the BB-3 machine, if it is in state q_1 and the head points to a W, is (B, q_2, R)—the machine writes a B on the tape, changes to state q_2, and moves to the right. This transition is shown in Figure 3. The operations required to bring the tape from an ID at the last timestep T, to an ID at time T+1 are given at the right of each tape. The machine halts when, if ever, it is in state q_3 and the head points to a B. On a *blank* tape of white symbols it takes 13 steps to print 6 black symbols and halt. While printing 6 black symbols is not a "useful" computation, this BB-3 machine is distinguished because no *smaller* Turing machine can do so.

[3] The Busy Beaver problem for a Turing machine with N states (BB-N) is the problem of designing a N-state Turing machine with two symbols, black and white, that prints the greatest number of black symbols before halting [**Ab**]. The example we give is a solution for the three state problem. If we consider the number of black symbols that are printed by the solution to the BB-N problem to be f(N) we can show that f(N) increases faster than any function computable by a Turing machine. f(N) has been determined for N up to N=6 but there is no algorithmic or "mechanical" way to find f(N) for any N. Functions like f(N) are said to be *uncomputable*. This is an example of the results that computer scientists prove with the Turing machine abstraction.

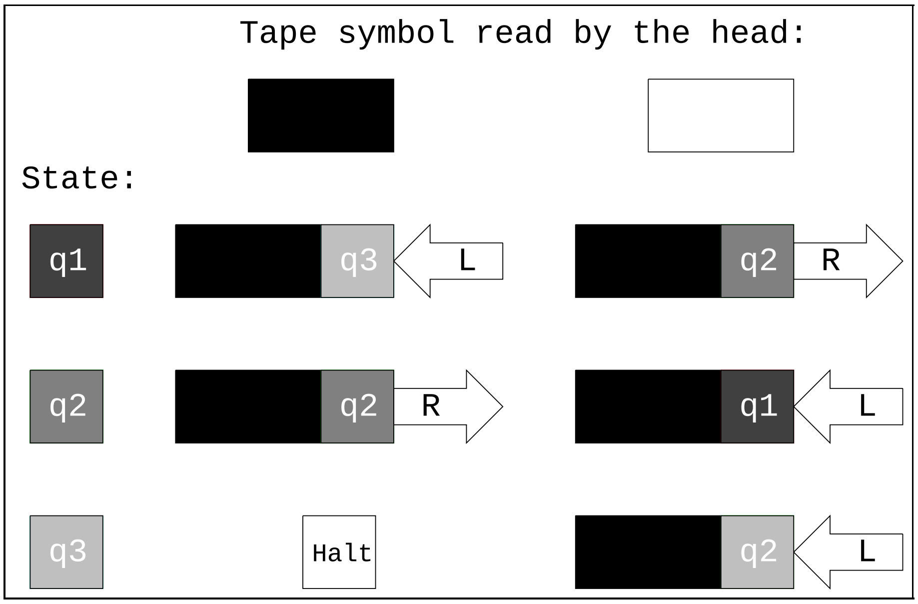

Figure 2: The transition table for a three state Busy Beaver machine.

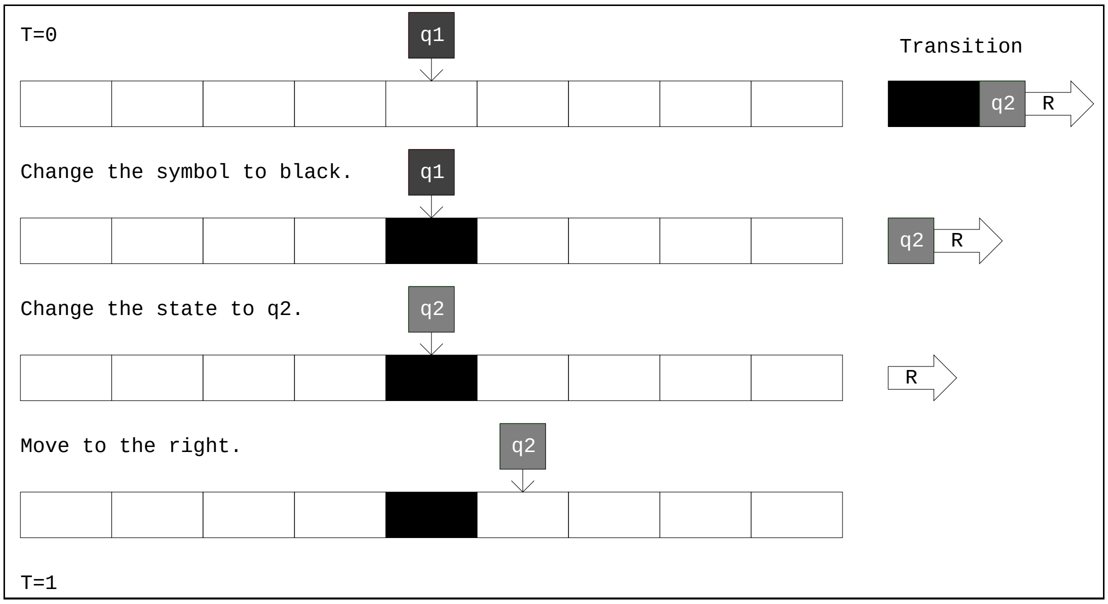

Figure 3: The first timestep of the simulation of the BB-3 on a blank tape.

1.4 Universal Turing machines

While a Turing machine may be constructed to implement any specific algorithm that one can imagine, it would be awkward to have to build a new physical machine every time one wanted to solve a new problem. One's desk would quickly fill up with machines dedicated to different computations. Fortunately, and amazingly, a Turing machine can be constructed that takes as an input a description of another Turing machine and a data tape, and simulates that Turing machine on its own tape. Such a Turing machine is known as a Universal Turing Machine (UTM). The personal computers that we use everyday are good approximations of Universal Turing machines—the programs that they run are descriptions of specific algorithms and hence specific Turing machines. Our personal computers only fall short of UTMs in that their memory cannot be expanded every time we need more storage. The presentation of a molecular computer with the power of a UTM is the goal of this paper.

2 DNA

2.1 DNA, structure and conventions

A single strand of DNA can be likened to a storage tape that can support a four symbol alphabet, S = {A, G, C, T} representing the nucleotides adenine, guanine, cytosine and thymine. These nucleotides are not bonded directly to one another but rather hang from a phosphate and sugar backbone. The DNA strand's backbone has a *polarity;* a sequence of DNA is distinct from its reverse. In order to represent the polarity we name the ends of a piece of DNA according to the structure of its phosphate backbone. One end is called the $5'$ end—its terminal phosphate is attached to the $5'$ oxygen of a sugar and the other end is called the $3'$ end—its terminal phosphate is attached to the $3'$ oxygen of the sugar.

Taken as pairs the nucleotides A and T and the nucleotides C and G are said to be complementary. This means that an A-T pair or a C-G pair can form weak, non-covalent bonds known as hydrogen bonds that serve to hold them together. When a stretch of single-stranded DNA encounters another stretch of single-stranded DNA that has a complementary sequence the hydrogen bond interactions between complementary pairs join the two strands in a process called *annealing.* A piece of single-stranded DNA will only anneal to its complement if it has the opposite polarity. When we look at a piece of double-stranded DNA one strand extends from $5'$ to $3'$ and the other from $3'$ to $5'$. In this paper we draw double-stranded DNA with the top strand oriented $5'$ to $3'$ and the bottom strand oriented $3'$ to $5'$.

2.2 DNA as a computing medium

DNA, to a computer scientist, looks like the tape of a Turing machine. The similarity has prompted others to think of it as a media for computation ([**Ad1**], [**Ben1**]). As such DNA has several attractive qualities:

(A) *DNA is the genetic material.*

DNA is the storage medium for genes—the plans for the protein molecular machines that perform most of the chemistry in all living things. Our genes determine our morphology, and influence our behaviour. Naturally we are interested in any methods that can modify them in a general way (*e.g.* Turing machines). We also know genes modify themselves with all manner of recombination events. These events may follow rules that allow us to identify them with computation or the generation of languages.

(B) *There are many enzyme-mediated chemical reactions on DNA.*

Nature provides us with a large "toolbox" of enzymes with which to manipulate DNA. These tools range from simple string catenation and string splitting operators like ligases and restriction enzymes to complex copying machinery like polymerases. We steal genes for these enzymes from a variety of organisms, clone them into easy growing bacteria like *E. coli* and harvest them for use in molecular biology.

(C) *DNA is small and easily copied.*

There are 67 atoms per A-T pair and 66 atoms per C-G pair.[4] DNA supports four symbols so this gives a capacity of 1 bit per 33 atoms for double-stranded DNA. The average molecular weight of one base pair is 660 Daltons [**Kor**]. This gives an impressive .33 kg DNA / mole bits.[5] In addition, DNA can be amplified very quickly with the polymerase chain reaction (PCR)— over millionfold in an hour [**Kor**].

(D) *Reactions between DNA species at equilibrium are completely reversible.*

Charles Bennett has proposed that computers based on DNA or a similar macromolecule would be good candidates for practical reversible computers. Normally computations are carried out in irreversible steps that lose information and dissipate heat. Because enzyme catalyzed operations between DNA species at equilibrium are completely reversible any computations that we embed in them are reversible too. At equilibrium the forward and reverse rates of a chemical reaction are the same. This means that any computation associated with the reaction moves backwards and forwards with equal rates as well; on average no progress is made. To drive the reaction/computation forward reactants must be added or products removed. The energy dissipated depends only on how fast one pushes the computation to proceed [**Ben2**]. In principle, one can spend as little energy as desired by slowing down the computation.

3 RESTRICTION ENZYME OPERATIONS ON DNA

"...the type II nucleases are clearly one of nature's greatest gifts to science."
—Arthur Kornberg in DNA Replication

To create a correspondence between DNA and Turing machines we needed to find some operations on DNA that could be made to correspond with the primitive operations of a Turing machine. We chose to explore the implementation of a DNA based Turing machine with the chemistry of restriction enzymes because we felt it rich and complex enough to do so. Operations on DNA with the most common class II restriction endonucleases proved to have undesirable properties that thwarted efforts to implement a Turing machine. A subgroup of the class II restriction endoncleases, the asymmetric or class IIS restriction enzymes, were found to have operations with properties that lent themselves more readily to Turing machine design.

3.1 Class II restriction endonucleases

Bacteria employ *restriction endonucleases* or *restriction enzymes* to cut double-stranded DNA at or near specific words known as *restriction sites* or *recognition sites.* These enzymes are used to chop up foreign DNA, like that from viruses, which enters the bacterium. The bacterium's own DNA is unaffected because the bacterium's own DNA is chemically modified at the recognition sites in such a way that the restriction enzyme cannot cut it.[6]

Most restriction enzymes recognize 6-8 base pair sequences of double-stranded DNA. These recognition sites have an inverted mirror plane in the middle so that the first half of the site is the reverse of the complement of the second half of the site. Below the inverted mirror plane is indicated by a |.

```
      5'          3'
.... ACTG | CAGT ....
     TGAC | GTCA
      3'          5'
```

Biologists call sequences with this kind of symmetry *palindromic.* The cuts made by an enzyme recognizing such a sequence are also made in an inverted mirror symmetric way. One of the most commonly used restriction enzymes of this type is *Eco*R I[7] that cuts at the $\bigtriangledown$[8] in the recognition site below:

[4] This includes the sugar-phosphate backbone and two Na^+ ions per base pair.

[5] Remember a mole is 6.02 x 10^{23}!

[6] The chemical tag—a methyl group—that protects the bacterium's own DNA is added to the recognition site by a modification enzyme known as a methylase [**Nai**].

[7] We name enzymes for the organisms from which we borrow them—in this case **E***scherichia* **co***li*.

[8] The positions at which an enzyme cuts are its *cleavage sites* or *cutting sites*.

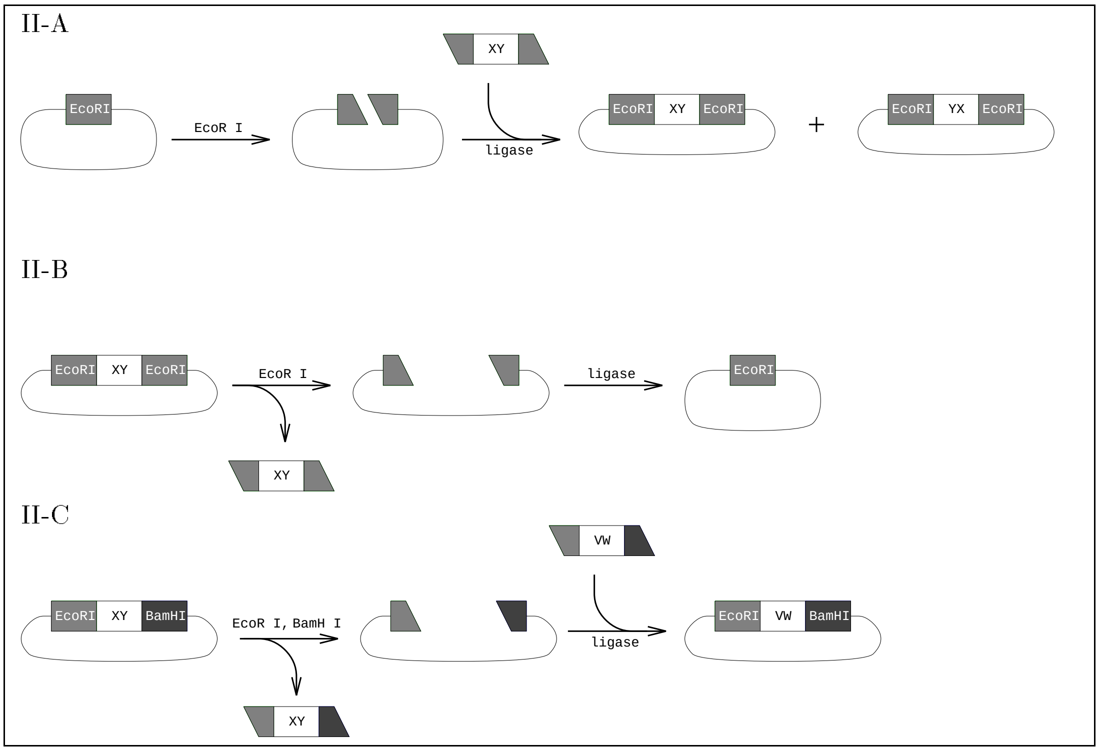

Figure 4: Operations using class II endonucleases.

```
 ....GAATTC....            ....G         +   AATTC....
     CTTAAG....   ------>  ....CTTAA              G....
```

Enzymes with this symmetry are known as class[9] II endonucleases or ENases-II of which over 1000 are known [**Sz1**].

We call the reaction of DNA with a restriction enzyme a *restriction digest.* A restriction digest in which one restriction enzyme is used is known as a *single digest*; if two or three enzymes are employed the reaction is known as a *double digest* or *triple digest* respectively. In practice the treatment of DNA with more than three restriction enzymes at once is rare since different restriction enzymes may cut DNA optimally under different reaction conditions.

3.1.1 Operations of class II restriction enzymes

A single-stranded overhang at the end of DNA cut by a restriction endonuclease can anneal to the single-stranded overhang of a different piece of DNA cut by the same enzyme—for this reason such single-stranded overhangs are referred to as *sticky ends.* When two ends have annealed another enzyme, DNA ligase, may be applied. DNA ligase repairs the cuts in the backbone and a continuous piece of double-stranded DNA is formed. This property of restriction enzyme cut ends allows the following operations on circular DNA molecules known as *plasmids*:

Operation II-A *Insertion of a DNA fragment into a plasmid without orientation control.*

A single restriction site (*Eco*R I) in a plasmid may be cut,[10] a fragment `XY` with two matching ends annealed, and the ends ligated (Figure 4, II-A). In this operation all of the overhangs are identical, the

[9] Biologists use “type” and “class” interchangeably when classifying restriction endonucleases.

[10] Here we represent the cut made by Eco RI in a schematic way that emphasizes that the “inverted mirror image” is identical to its mate and hence both sticky ends are identical and self-complementary.

fragment is nonorientable, and the reaction yields two products.

Operation II-B *Deletion of a fragment from a plasmid.*

The product of an insertion operation is the substrate or "input" for a deletion operation. A fragment can be deleted if it is flanked by two restriction sites for the same enzyme (Figure 4, II-B). The solution can be diluted until conditions favor circularization and the plasmid DNA can be recyclized without the fragment.[11]

Operation II-C *Replacement of a fragment in a plasmid with orientation control.*

If a second restriction site, say BamH I, replaces one of the restriction sites normally used in a deletion operation then a double digest using *Eco*R I and BamH I excises an oligonucleotide `XY` with two different sticky ends. We can then ligate in a fragment `VW` that has two different ends, assured that its insertion will be oriented (Figure 4, II-C).

3.1.2 Problems with class II restriction enzymes

While the chemistry of class II restriction endonucleases gives us a number of useful operations on DNA strings, several problems[12] are encountered if we try to use them to construct a Turing machine:

Problem 1 *One unique sticky end per restriction enzyme.*

Because the cleavage of a class II Enase occurs within the recognition site each enzyme generates one and only one kind of overhang. Here *kind* is defined by three variables—the length, sequence, and polarity ($5'$ or $3'$) of the overhang. This may seem more of an "obvious property" of class II restriction enzymes than a factor limiting their use but for simple DNA encodings of Turing machines it makes the number of restriction enzymes required unmanageable. If we imagine a DNA encoding of a Turing machine that uses one insertable oligonucleotide to represent each state–symbol transition then we must use one unique sticky end and hence one restriction enzyme for each entry in the transition table. Even the implementation of the smallest UTM known—a 4 symbol, 7 state machine [**Mi**]—would require 24 different restriction enzymes and many multiple digests to effect each transition. In the Section 3.2 we will describe restriction enzymes that do not suffer from this "one enzyme-one end" limitation.

Problem 2 *Orientability and the problem of Palindromic ends.*

The identical ends which caused the orientability problem we encountered in Operation II-A above hint at greater problems for symmetric class II Enases. Palindromic sticky ends like those generated by class II Enases are always self-complementary. This means that side reactions, other than those described in Operation II-A, will occur. The large plasmids can, after being cut, ligate together to form larger circular dimers and trimers (Figure 5, A). Fragments meant for insertion will ligate to themselves forming multimeric repeats known as *concatemers* (Figure 5, B). Using two restriction sites, as described in Operation II-C, decreases the number of unwanted end matchings in half but will still result in concatemers—the orientation of fragments will just alternate (Figure 5, C). The side reactions resulting from the use of palindromic restriction sites greatly increase the amount purification required between steps and waste reagent oligonucleotides.

Problem 3 *Regeneration of restriction sites*

[11] It is interesting to note that this deletion operation is really the same operation as insertion—only the relative concentrations of insert and plasmid DNA differ at the time of ligation.

[12] All of these "problems" are actually just artifacts of the way we choose to limit our use of restriction enzyme chemistry. For example, restriction sites whose subsequences are prefixes for other restriction sites serve as the basis for a whole host of other restriction enzyme operations which can solve all of the problems we list for class II restriction enzymes. The overhangs created by cutting sites with these "partial overlaps" can be chewed up, filled in, and ligated in a variety of ways to destroy the old restriction site and create a new one. Molecular biology, as practiced in lab, is really a collection of just such "clever hacks." To make our restriction enzyme operations independent of the use of specific restriction enzymes we have avoided the use of clever hacks wherever possible.

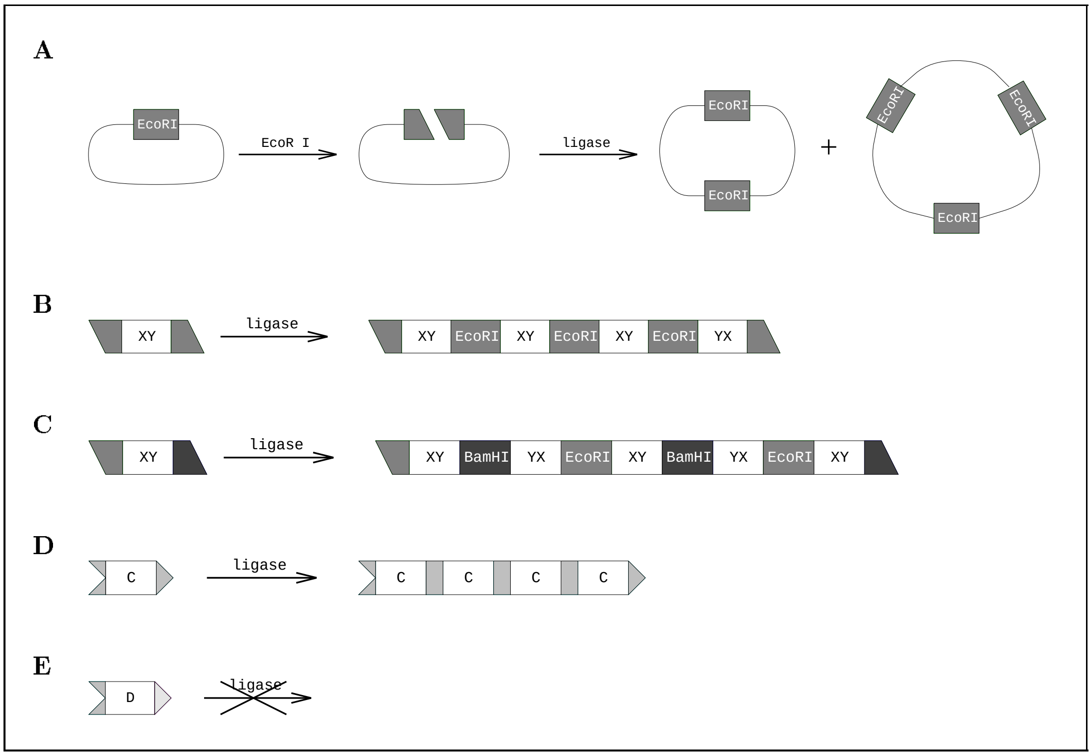

Figure 5: Concatemerization of oligonucleotides.

Because the cleavage sites of Class II Enases occur within their recognition sites the ligation of the overhangs they create results in the regeneration of the original restriction sites. Once a restriction site is used in a plasmid it must be deleted before being used at a different place in that plasmid.

In order to remove an occurrence of a restriction site it must either be flanked by two identical restriction sites as in Operation II-B or be flanked by two different restriction sites as in Operation II-C. These restriction sites are themselves regenerated when the DNA is recircularized and must themselves be flanked by other sites if they are to be removed. This sort of "infinite regression" problem that requires the the use of more and more distinct restriction sites may be solved in several ways but they require more than the chemistry of restriction and ligation.

Problem 4 *Restriction sites "bound" the computation.*

The region in a plasmid that is "accessible" by restriction enzyme chemistry is exactly the region "bounded" by the two most distant restriction sites. While "new tape" can be added at the bounding restriction sites by insertion, each such operation adds a new occurrence of the bounding restriction site—since it is no longer unique it cannot be used as a site of insertion. In order to access a larger amount of DNA, more distinct restriction sites must be added "outside" of the existing ones. This leads to the same sort of infinite regression problem inherent in the deletion of restriction sites makes designing tapes of arbitrary length difficult.

3.2 Class IIS restriction endonucleases.

Because class II Enases have recognition sites which are regenerated when fragments are ligated, because they generate palindromic sticky ends which increase side reactions, and because each enzyme can only generate one unique sticky end it seems that they are not good candidates for implementing a Turing machine.

Fortunately, a subgroup of class II restriction enzymes, the class IIS restriction enzymes do not recognize palindromic recognition sites and they cut far away from their restriction sites. One example of a class IIS enzyme is *Fok* I which cuts:

```
                      ▽
....GGATGNNNNNNNNNNNNNNNN....   ---->   ....GGATGNNNNNNNNN      +      NNNNNN....
....CCTACNNNNNNNNNNNNNNNN....           ....CCTACNNNNNNNNNNNNN             NN....
                           △
```

Where N $\in$ {A, C, G, T}, independently. These restriction endonucleases are also known as *nonpalindromic* or *asymmetric* restriction enzymes. The asymmetry of class IIS restriction enzymes allows the following new reactions:[13]

Operation IIS-A *Insertion of a DNA fragment with orientation control.*

Since class IIS enzymes cut away from their recognition sites in a stretch of arbitrary DNA one can choose to create an overhang that is non-palindromic. This means that an oligonucleotide `C` with ends matching the linearized plasmid can be inserted in only one orientation (Figure 6, IIS-A).[14] Here it is easy to see that one restriction enzyme is capable of creating many different sticky ends by varying the arbitrary 4 nucleotide sequence cut by *Fok* I. This solves the "one end-one enzyme" Problem 1 above but concatemers may still be formed (Figure 5, D) and the restriction site persists after the operation.

Operation IIS-B *Deletion of a fragment.*

Simple deletion of a fragment (Figure 6, IIS-B) parallels Operation II-B.

Operation IIS-C *Replacement of an oriented fragment.*

A pair of class IIS Enase sites with opposite directions can be used to prepare two overhangs derived from different sequences (Figure 6, IIS-C.) Now neither the insertion oligonucleotides (Figure 5, E), nor the cut plasmid DNA have self complementary ends and the desired insertion competes with many fewer side reactions. This solves problems Problem 2 but the restriction sites still persist after the operation.

Operation IIS-D *Deletion of a fragment with auto-excision of restriction sites.*

Back to back occurrences of class IIS restriction enzymes (Figure 6, IIS-D) can cut themselves out of a plasmid and the plasmid can be rejoined without the regeneration of a restriction site described in Problem 3.[15]

Operation IIS-E *Replacement of an oriented fragment with the excision of restriction sites.*

This operation is a straightforward combination of strategies used in Operations IIS-C and IIS-D. Here two back to back restriction sites are used to prepare overhangs derived from different cleavage site sequences. This operation solves both Problems 2 and 3.

Operation IIS-F *"Progress"—movement of a sequence through a strand of DNA.*

---

[13] These and many other operations possible with class IIS Enases are presented in an excellent review article by Szybalski *et al.* [**Sz1**] One class IIS operation not shown here that may find use in DNA computation is Syzbalski's "universal restriction enzyme", a special adaptor oligonucleotide that recognizes an arbitrary sequence of single-stranded DNA and directs a class IIS enzyme to cut it at an arbitrary position. [**Sz2**]

[14] In this schematic the recognition site of *Fok* I is represented by the `Fok I` labeled arrow while its cutting site is represented by the longer "arm." Here the ends created by *Fok* I are drawn in way that emphasizes that the "inverted mirror image" of a given end is not identical to its mate and is not self-complementary.

[15] It is interesting to note that this operation provides the proof that a pair of class IIS restriction sites can simulate *any* class II restriction site. To simulate any class II enzyme `f` which cuts a restriction site `F`, all we need to do is replace each occurrence of `F` with the string `FGG'F` where `GG'` represents a pair of back to back restriction sites for a class IIS enzyme `g` which recognizes `G` and will cut in the `F` seqences to create the same kind of overhang that `f` does when it cuts.

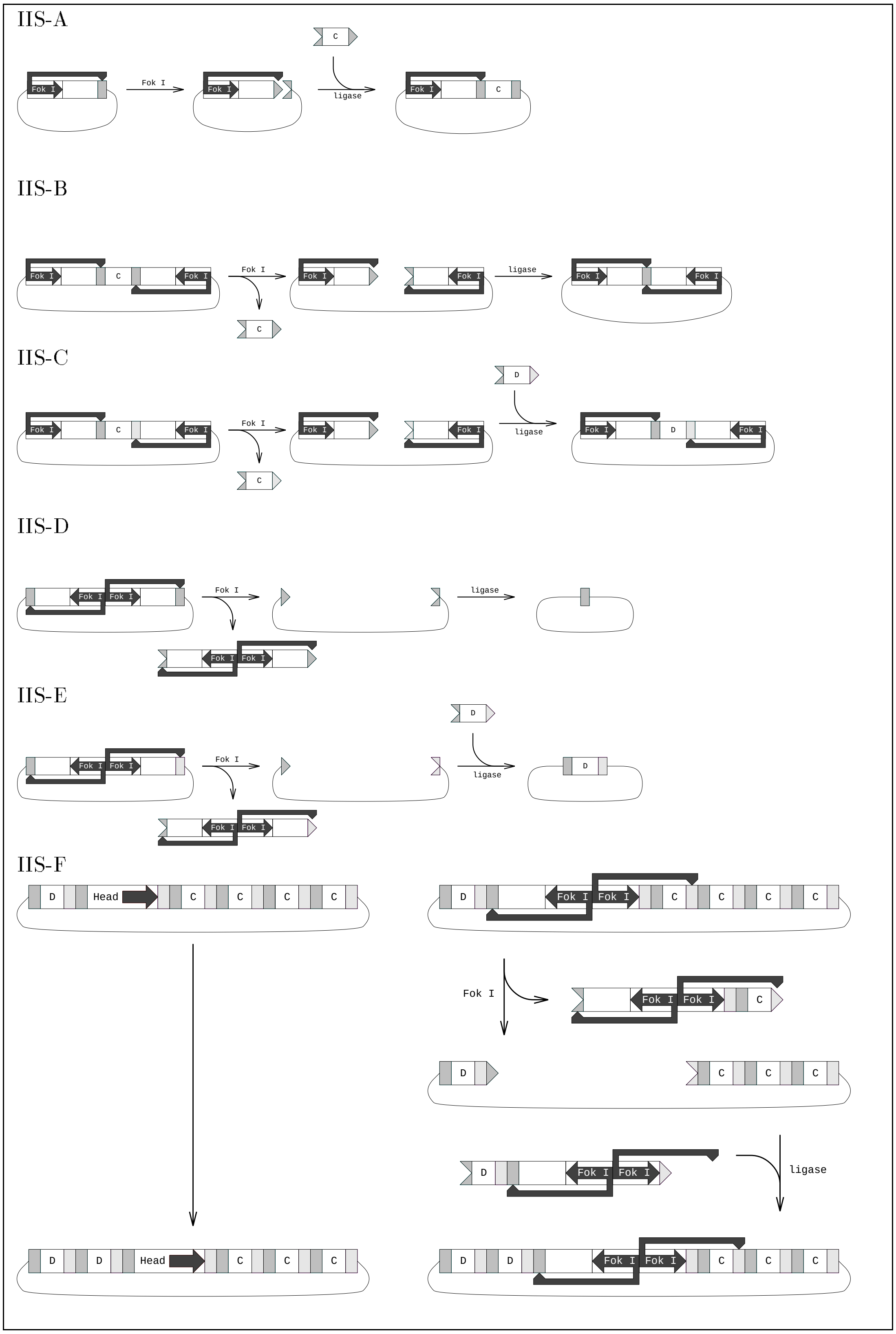

Figure 6: Operations using Class IIs endonucleases.

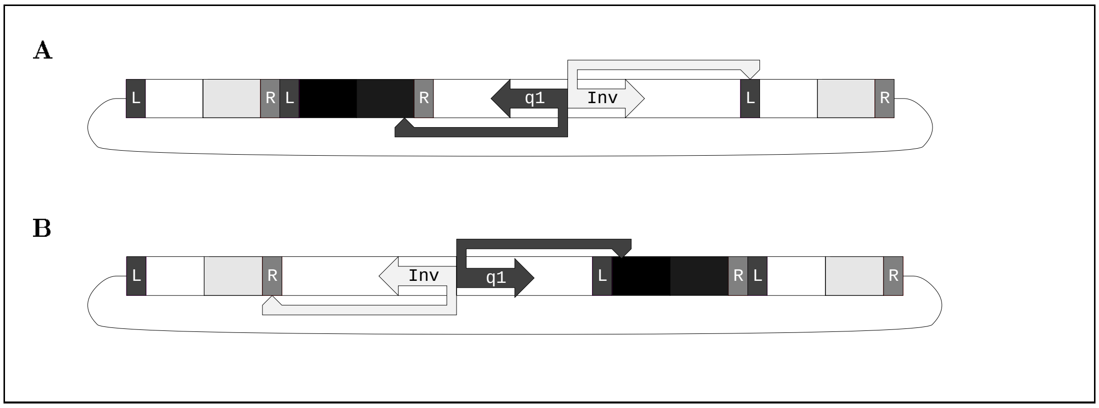

Figure 7: An instantaneous description of a TM in which the tape holds the string `WBW`, the head points at the `B` symbol, and the machine is in state `q1`.

The left half of Figure 6, IIS-F shows what we mean by "progress"; a sequence `Head` moves to the right through a string of `C` sequences and replaces them with `D` sequences.[16] The right half of the figure demonstrates how two back to back occurrences of *Fok* I could be used to move just such a `Head` sequence. This operation is really just a variation of Operation IIS-D in which a pair of back to back restriction sites have been added to the right of sequence `D` in the insert subject to the constraint that the rightmost recognition site has a cleavage site that does not lie entirely in the insert.

This operation highlights the ability of asymmetric restriction endonucleases to "reach out" and cut DNA away from their recognition site making regions of DNA "outside" of the two most distant restriction sites "accessible"—solving Problem 4.

The class IIS restriction endonucleases, then, can be used to solve all of the problems outlined in Section 3.1.2.[17] In the next section we describe how to use their operations to implement a Turing machine.

4 A DNA SCHEMATIC FOR THE BB-3 TM.

To organize the explanation of our DNA schematic we recall one natural division of the elements of a Turing machine: variable elements, and constant ones. First, the variable elements are represented by a single circular DNA molecule; next, the transition table is encoded by oligonucleotide inserts; finally, six chemical steps are described which apply the oligonucleotide encoded transition table to advance the Turing machine one timestep.

4.1 Encoding an instantaneous description

Figure 7 shows the two ways we encode an instantaneous description of the BB-3 TM in which the tape holds the string `WBW`, the head points at the symbol `B`, and the machine is in state `q1`. The representation of symbols, head position, and machine state, as well as the reason we allow two versions of any particular ID, are explained below.

[16] Not unlike the head of a Turing machine!

[17] Another subgroup of the class II restriction enzymes, those that recognize "interrupted palindromes", have cut sites in arbitrary stretch of DNA between the two halves of what would otherwise be a palindromic site. This means that they do not suffer from Problems 1 and 2 but because their cleavage sites are inside of their recognition sites they cannot solve Problems 3 and 4 as do the class IIS enzymes, without the use of overlapping restriction sites.

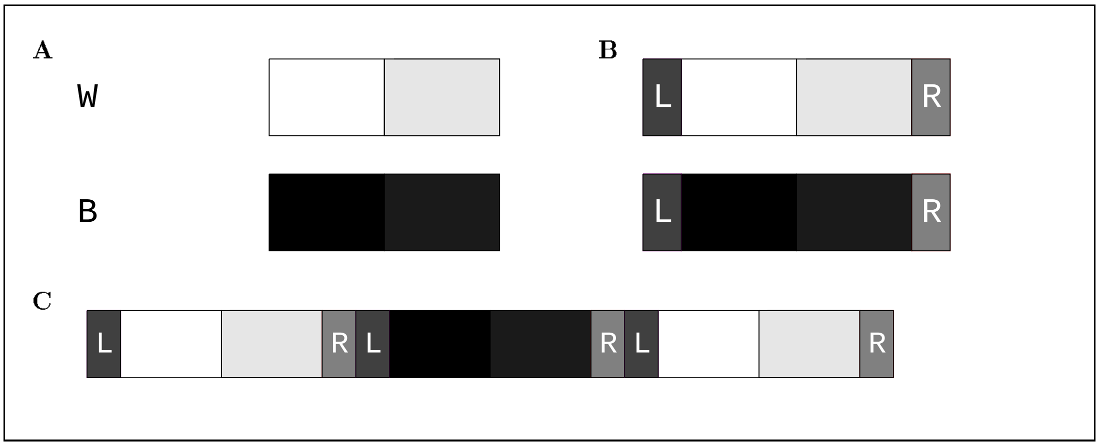

Figure 8: Schematic encoding of the black and white symbols.

4.1.1 Symbols

Two distinct DNA sequences are used to represent the symbols `W` and `B` (Figure 8, A). Each is subdivided into a left and right half. The DNA Turing tape as shown in Figure 7 is not just the simple catenation of `B` and `W` sequences. To each symbol we append, on the left, a short sequence labelled `L`, and, on the right, a short sequence labelled `R` (Figure 8, B). Because they are the same in both the `B` and `W` symbols these sequences are called the left and right *invariant* sequences. The DNA sequence for a Turing tape (without the head) holding the string `WBW` is shown in Figure 8, C.

4.1.2 The head

The two back to back asymmetric restriction sites labelled `Inv` and `q1` in Figure 7 represent the head of the Turing machine. The restriction site labelled with the state always points to the current symbol and `Inv` always points at an adjacent invariant sequence. Our encoding generates two DNA representations for any particular ID because we place the head of the Turing machine "inside" of the tape; one representation (A) positions the head sequence to the right of the current symbol, and the other (B) positions the head sequence to the left. The physical interpretation of these two possibilities is this: if the head is to the right of the current symbol then the last move of the machine was to the left; contrariwise, if the head is to the left of the current symbol then the last move of the machine was to the right.

4.1.3 The state

It is the *spacing* between the recognition site labelled `q1` and the current symbol that encodes the state of this Turing machine. Consider the 6 base pair oligonucleotide `W'`, the first half of the symbol `W`, shown below:

W' = **GAAACA**
CTTTGT

The 4 base pair cutting region of *Fok* I may be used to cut sequence `W'` in any one of three different cutting *frames* by varying the number of intervening bases between the *Fok* I recognition site and the symbol sequence (Figure 9).

We associate the frame in which a symbol sequence is cut with the concept of state in a Turing machine and call the enzyme used to cut symbol sequences in different frames a `state` enzyme or `state` cutter. Each half of a symbol sequence is large enough to be cut in three frames by `state` cutter enzyme. By carefully picking different DNA sequences for each symbol, the sticky end generated by cutting a given symbol in given reference frame can be made unique. Figure 10 shows the overhangs generated when the

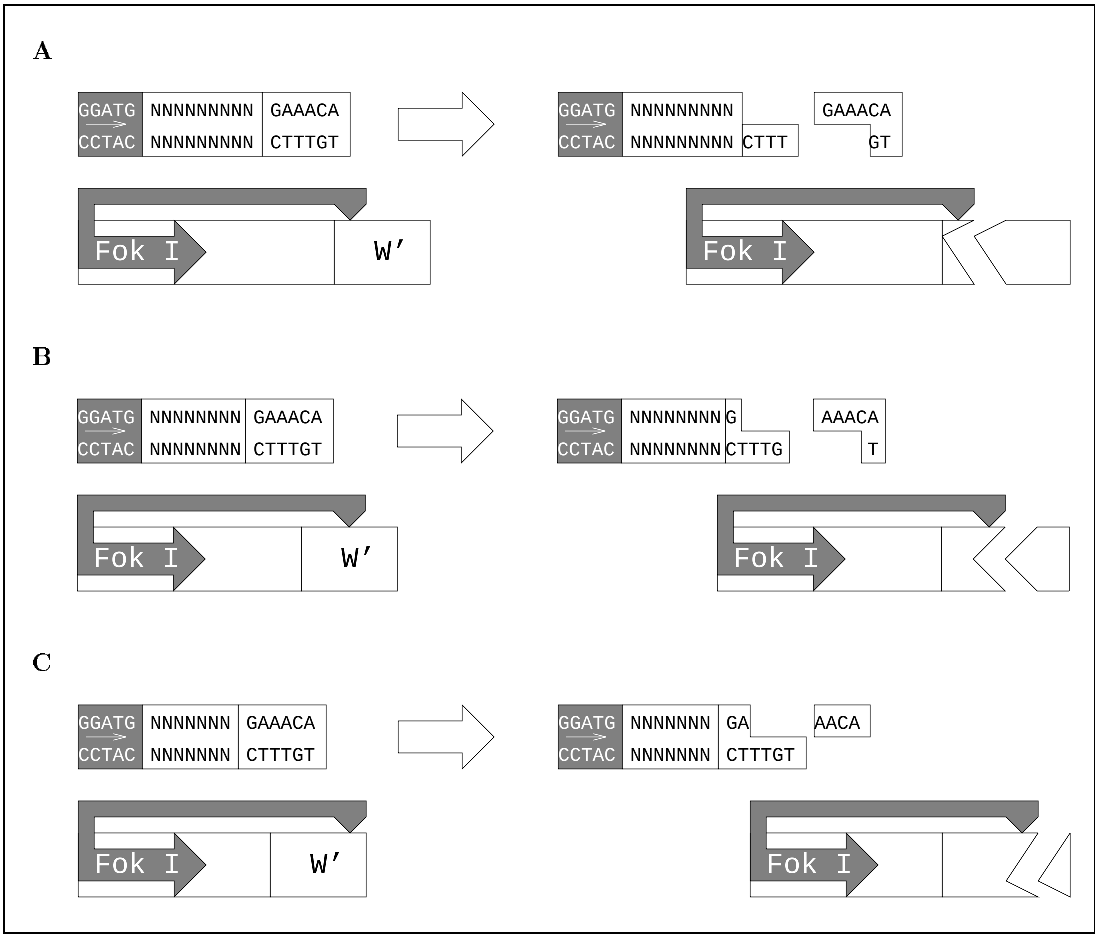

Figure 9: DNA and schematic representations of *Fok* I cutting a six base pair sequence, `W'`, in each of three possible frames. Note that the shape of the cut in the schematic representation differs in each reference frame—the closer the recognition site of *Fok* I to `W'` the higher the "notch" in the schematic sticky end.

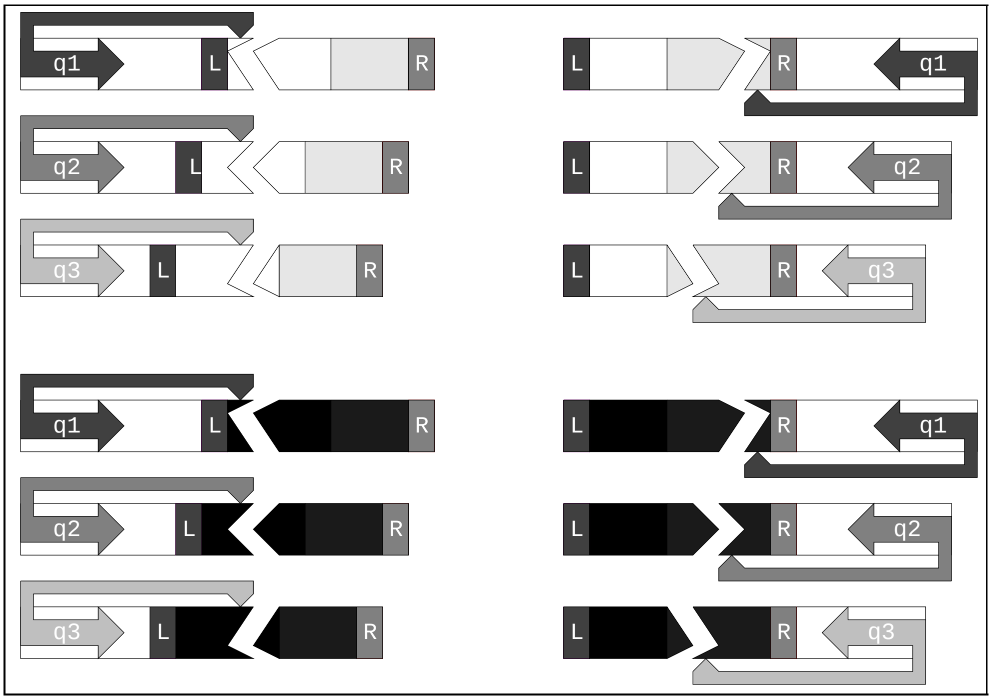

Figure 10: Overhangs generated when black and white symbols are cut by the `state` cutter in 3 different state frames from the left and from the right.

symbols `W` and `B` are cut in each of the three frames from the right and from the left. The restriction site for each frame is labelled with the state it represents and shaded to match Figure 2.

4.2 Encoding the transition table

The unique sticky ends which can be generated by cutting a symbol with `state` cutter allow us to ligate, into our DNA tape, a *transition oligonucleotide* which encodes the new state, symbol, and direction of the Turing machine. At any given timestep a Turing machine's last move may have been to the right or left, and its next move may be to the right or left. This means that there are four basic plans for transition oligonucleotides (Figure 11). Their parts are described below:

`Coh` is the cohesive end matching the end generated by cutting a given symbol with the `state cutter`.

`Sta` is a restriction site for the `state` restriction enzyme. It cuts the current symbol according to the state of the machine.

`Em` is the restriction site for the `end-maker` restriction enzyme. It cuts to create the same size and orientation overhang as the `state` restriction enzyme but it has a distinct recognition site. This site allows the cohesive end to be prepared by cleavage with the `end-maker` enzyme and is used *only* in the manufacture of the transition oligonucleotides.

`Res` for "result" is a sequence encoding the new symbol.

`L` and `R` are two distinct "invariant" DNA sequences that separate symbol pairs in the DNA sequence.

`Cap`, `X`, and `Inv` are class IIS restriction sites whose enzymes all cleave `L` or `R` to give the same size and orientation overhang. These sites are used to cleave `L` or `R` sequences at various stages of the computation. `Cap` is recognized by the `cap` enzyme, `Inv` by the `invariant` enzyme, and `X` by the `symbol-excision` enzyme.

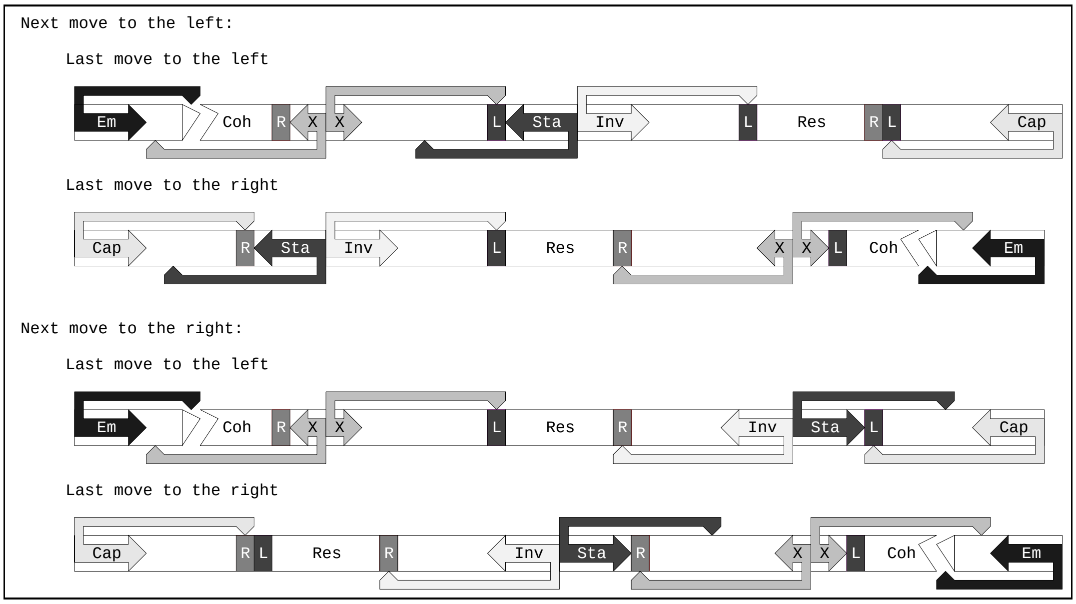

Figure 11: The four forms transition oligonucleotides take. After being cleaved with `end-maker` enzyme to remove the `Em` sequences, each transition oligonucleotide can be divided into 5 sections: `Coh`, the sticky end specific to a particular state-symbol combination, the head sequence (`Inv` + `Sta`), the result region `Res`, the symbol-excision sites `X`, and the `Cap` sequence. `Coh` matches a symbol cut during the last timestep by `state cutter` enzyme. If the machine's last move was to the left, `Coh` is the left end of the transition oligonucleotide and vice versa. The symbol-excision sites `X` are always placed next to the sticky end `Coh`. `Cap` is placed on the opposite end of the oligonucleotide. Between the `X` and `Cap` sequences we place the head and `Res` sequences. The `Sta` site in the head sequence points in the direction the head sequence will move. The result region is always placed "behind" the head sequence, that is, next to `Inv`.

We translate the transition table for the Busy Beaver machine given in Figure 2 using these four forms, making one oligonucleotide with a sticky end `Coh` to match each end in Figure 10. This generates the the oligonucleotide encoded transition table give in Figure 12.[18] A j x k TM encoded this way has $2jk$ transition oligonucleotides.

4.3 Making a transition—Going from one ID to the next.

An instantaneous description for BB-3 machine operating on a blank tape is shown in Figure 13. The series of 6 distinct chemical steps required to take this ID at time T to the ID at time T+1 are shown in Figure 13 and are explained below:

Steps 1-4 replace the head with the correct transition oligonucleotide in an process analogous to Operation IIS-E.

1. Cut the current symbol with the `state` and `invariant` restriction enzymes.[19] The `state` cutter creates an end unique to the current symbol and the current state. The `invariant` cutter cleaves an `R` sequence to create an end that is the same regardless of what symbols lie to the left of the computation.[20]
2. Mix the twelve transition oligonucleotides in Figure 12 with the DNA Turing tapes.[21] Operation IIS-E assumes that the oligonucleotide insert `D` is has unique cohesive ends on both ends. Because only the sticky end generated by the `state` cutter is unique we keep the sticky end on the transition oligonucleotide that matches the end generated by the `invariant` cutter protected by the `Cap` sequence. Use DNA ligase to join the transition oligonucleotide to the tape.
3. Cleave the `Cap` sequence that protects the invariant sequence `R` on the oligonucleotide.[22]
4. Circularize the DNA with DNA ligase. Incorrect ligations can occur if the left cohesive end of an invariant sequence on one DNA tape sticks to the right cohesive end of an invariant sequence on another DNA tape. The reaction can be run at very dilute concentrations of DNA so that intramolecular closure is favored.

Steps 5 and 6 serve to delete the previously read symbol from the tape using the concept of "progress" developed in Operation IIS-F.

5. Cut with `symbol-excision` restriction enzyme. This cuts away the previous symbol and leaves two matching invariant ends.
6. Recircularize with DNA ligase to join the invariant ends created in 5. As in in 4, reaction conditions should be adjusted to favor intramolecular closure. The `state` cutter restriction site now "points" to the current symbol and the DNA tape once again represents an instantaneous description.

Steps 1-6 are repeated until a `Halt` is incorporated and the computation is done. After any step 6 the computation may checked for the incorporation of a `Halt` seqence. This can be done by PCR amplification of a small aliquot of the computation using `Halt` sequence as a primer. Only halted tapes would be amplified—if detected they could then be sequenced to recover the answer. This scheme is fine if all of the DNA tapes

[18] Note that the "whitespace" between a given `state` cutter recognition site and the adjacent invariant sequence `R` or `L` differs between an oligonucleotide encoding a change of direction (*e.g.* moving from the left going to the left) or an oligonucleotide encoding a preservation of direction (*e.g.* moving from the left, continuing to the right). If the Turing machine keeps moving in the same direction then there are will always be two invariant sequences between the `state` sequence in the head and the current symbol seqence. If the Turing machine changes direction then for the next move there is only one invariant sequence between them. In our schematic an invariant sequence is the width of 1 cutting frame so the "whitespace" in the transition oligonucleotides has been adjusted accordingly.

[19] This may be performed as a double digest or two sequential digests if the `state` and `invariant` enzymes require buffers that are too different.

[20] If the last move had been from the right the `invariant` cutter would have cleaved an `L` sequence instead.

[21] Remember that these transition oligonucleotides have already been treated with `end-maker` restriction enzyme which cuts at site `Em` to create their unique sticky ends.

[22] Some restriction enzymes have difficulty cutting restriction sites near the ends of oligonucleotides. For this reason a few arbitrary nucleotides (not shown) may have to be added to the end of the `Cap` sequence.

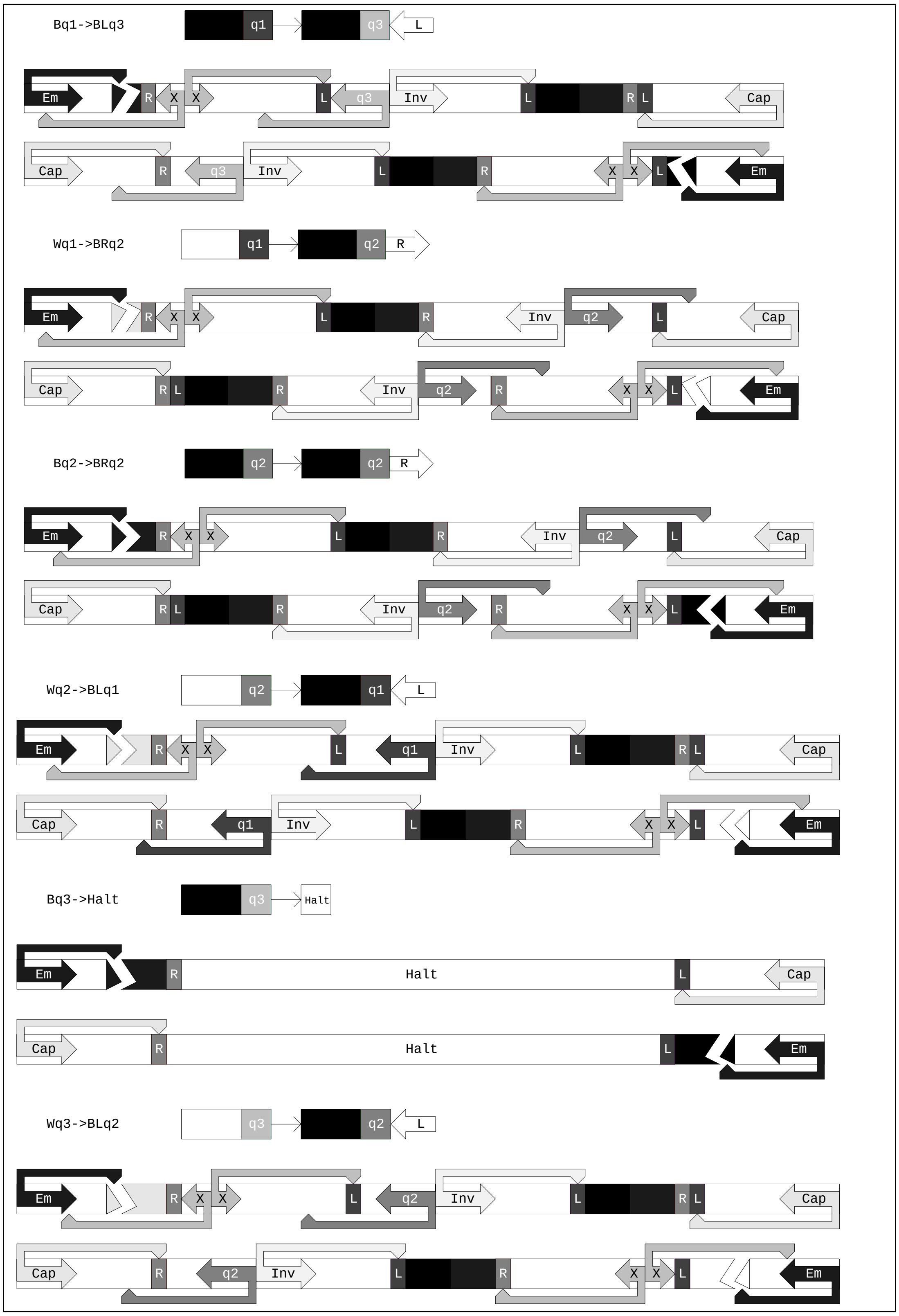

Figure 12: The 12 oligonucleotides which encode the BB-3 machine transition table. Note that in each pair of oligonucleotides the top oligo matches a symbol cut from the left and the bottom oligo matches the same symbol cut from the right.

Figure 13: The first step of the BB-3 machine simulated on a blank tape. **0** encodes the first ID. **1** shows the result of cleaving the DNA tape with `state` and `invariant` enzyme. The white cohesive end is specific for the symbol-state combination (`W`, `q1`) and matches an oligonucleotide encoding the transition `Wq1 -> Bq2R` shown, in schematic form, to the right of the tape. **2** shows the ligation of the transition oligonucleotide encoding the transition `Wq1 -> Bq2R` to the unique sticky end created in **1**. In **2**, note that the direction of `q2`'s "cutting arm" determines the direction of the head, the spacing between `q2` and the invariant sequence `R` encodes the new state, the black sequence "behind" the head encodes the new symbol and the white sequence to the right of the `X` recognition site encodes the last symbol. **3** shows the cleavage of the `Cap` sequence protecting the invariant sequence `R`. **4** shows the intramolecular closure of the DNA tape. **5** shows how treatment with `symbol-excision` enzyme effects the excision of the last symbol. **6** shows the cyclization of the tape to form the next ID.

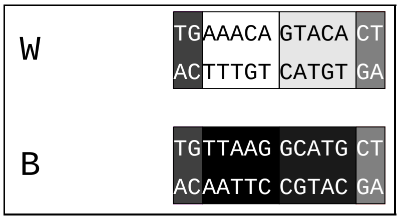

Figure 14: Black and white symbols expressed as real DNA.

Em:	Bbv I	GCAGCNNNNNNNN NNNNNN CGTCGNNNNNNNNNNNNN NN	Cap:	BsrD I	GCAATGNN NN CGTTAC NNNN
Sta:	Fok I	GGATGNNNNNNNNN NNNNNN CCTACNNNNNNNNNNNNN NN	X:	Bpm I	CTGGAGNNNNNNNNNNNNNNNN NN GACCTCNNNNNNNNNNNNNN NNNN
Inv:	BseR I	GAGGAGNNNNNNNNNN NN CTCCTCNNNNNNNN NNNN	Sta′:	Hga I	GACGCNNNNN NNNNNNN CTGCGNNNNNNNNNN NN

Figure 15: The restriction enzymes used in a DNA Turing machine, their restriction sites, and the overhangs they generate. **Em**, **Sta**, **X**, **Inv**, and **Cap** are used to implement the BB-3 machine. **Sta'** is an additional enzyme used to implement Minsky's 4 x 7 UTM.

perform the same computation but if different tapes encode different problems some will be lost every time the computation is checked for `Halt` sequences. If a biotin label[23] were incorporated into the `Halt` sequence then streptavidin coated beads could be used to recover halted machines before the PCR amplification step. Only halted tapes would stick to the streptavidin beads so no portion of the unfinished tapes would be removed.

5 REAL DNA SEQUENCES FOR THE BB-3 TM.

We now give real sequence and restriction site assignments to the schematic representation presented in Section 4. First, the invariant sequences, `R` and `L`, and the symbols `W` and `B` are assigned in Figure 14. Next, the restriction sites for the enzymes, *Bbv* I, *Fok* I, *Bse*R I, *Bsr*D I, and *Bpm* I are assigned to the **end-maker**, **state**, **invariant**, **cap**, and **symbol-excision** sequences (Figure 15). The unique sticky ends generated by *Fok* I cutting the symbol sequences in each of three different frames are given in Figure 16. Each overhang is designed so that it is complementary to no other overhang (or complement of an overhang) at more than two positions. This constraint seems to work well in gene construction [**Nas**]. The transition table oligonucleotides are constructed by a direct substitution of the chosen symbol sequences, restriction enzyme sequences and cohesive end sequences into the schematics in Figure 12.[24]

6 CONSTRUCTING A DNA UTM.

Given enough restriction enzymes, with long enough overhangs and large enough sequences between their recognition and cleavage sites, any Turing machine could be implemented with our technique. In reality, however, the number and variety of class IIS restriction enzymes is limited so only small Turing machines may be constructed. To show that restriction enzyme chemistry is universal, without the design of imaginary enzymes, we demonstrate that our model can implement the smallest UTM reported, a 4 symbol, 7 state

[23] Biotin is just a functional group that binds strongly to another fuctional group known as streptavidin.

[24] Actually, an extra 2 nucleotide spacer has to be added between the symbol excision sites `X` and the sticky end `Coh` because *Bpm* I excises a longer intervening sequence than is necessary to cut out the last symbol.

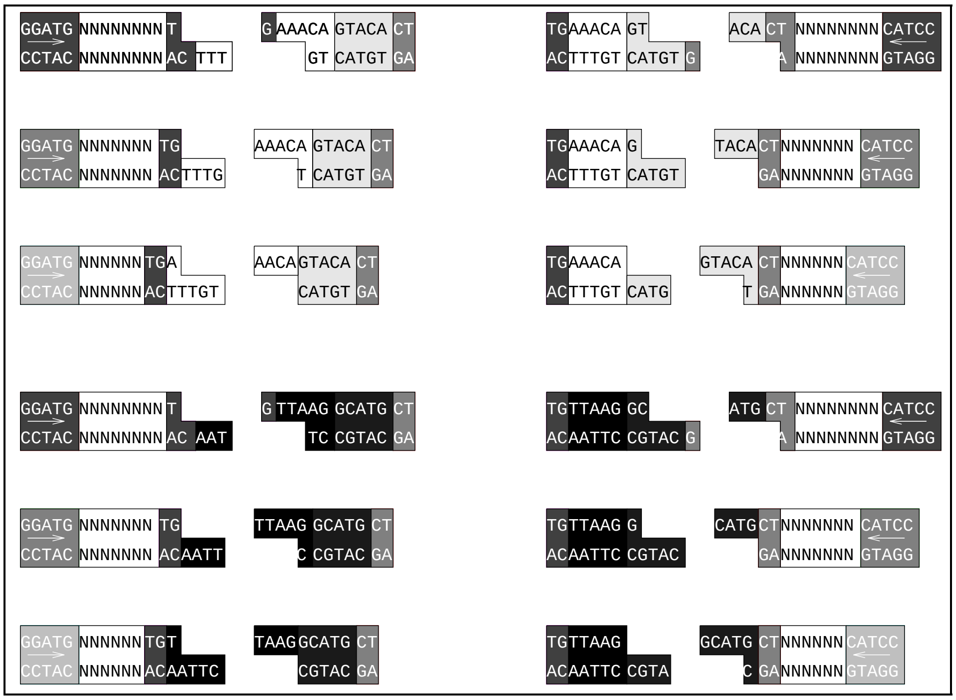

Figure 16: Overhangs generated when black and white symbols are cut by the **state** cutter *Fok* I in 3 different state frames from the left and from the right. Each overhang is unique and mismatches every other overhang (or complement of an overhang) in at least two positions. Note that the overhangs for state `q1` actually include 1 nucleotide of the 2 nucleotide invariant sequences. This is alright—the invariant sequence is regenerated when the correct transition nucleotide is added. This use of part of the invariant sequence as symbol sequence allows us to make the symbols shorter.

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
Y	OL q_1	OL q_1	YL q_3	YL q_4	YR q_5	YR q_6	OR q_7
O	OL q_1	YR q_2	Halt	YR q_5	YL q_3	AL q_3	YR q_6
I	IL q_2	AR q_2	AL q_3	IL q_7	AR q_5	AR q_6	IR q_7
A	IL q_1	YL q_3	IL q_4	IL q_4	IR q_5	IR q_6	OR q_2

Figure 17: The transition table for Minsky's 4 x 7 Universal Turing machine.

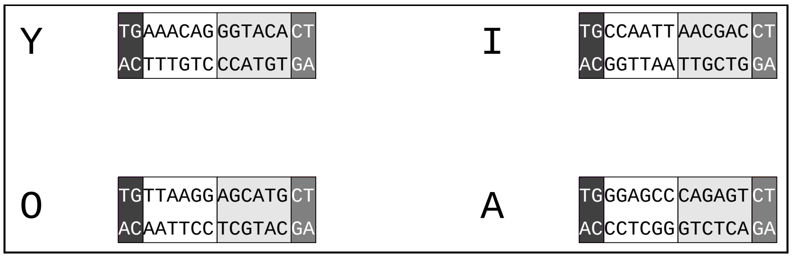

Figure 18: 4 DNA sequences used as the symbols for a UTM.

machine constructed by Minsky (Figure 17). This machine is constructed using the 4 symbols shown in Figure 18. These symbols have the same property as those chosen for the Busy Beaver machine. Every four base overhang created by a `state` restriction enzyme mismatches ever other such four base overhang in at least two places.[25] *Fok* I's cutting site can only be shifted through these symbols 4 times to yield 4 different states. To implement the remaining 3 states, an additional enzyme, *Hga* I, is used as `state` cutter. *Hga* I makes 5 nucleotide overhangs (see restriction site `Sta'` in Figure 15) that can be shifted through 3 cutting frames in the given symbols. Because the overhangs *Hga* I creates contain the 4 symbol sequences used in the *Fok* I overhangs for states 1–3 as subsequences they also have the property that any pair of mismatched ends will have at least 2 mismatches. A UTM so constructed has 4*7*2 = 56 transition oligonucleotides. Unfortunately, no counterpart to the `end-maker` *Bbv* I exists for *Hga* I, that is, there is no 5 base overhang cutter that recognizes a different site from *Hga* I for use in preparing the end of those oligos that anneal symbols cut by states 5, 6, and 7. In order to manufacture them we must either synthesize both strands explicitly[26] or synthesize them as two parts, one with a pre-cut *Hga* I restriction site, and the other with an uncut *Hga* I restriction site, that may be ligated together.

7 DISCUSSION

7.1 Error correction

There are many ways for the 6 chemical steps outlined may fail. We have listed the most important error modes and strategies for their minimization below:

[25] The sequences for the symbols `W` and `B` are really just the sequences for `Y` and `O` less their central two nucleotides.

[26] For the other transition nucleotides which use *Fok* I sites to encode state we need synthesize only one strand and fill in the other with DNA polymerase. Transition oligonucleotides that may be manufactured in this way have the added advantage of being able to be cloned into plasmids or copied by PCR.

(A) *Failed ligations*

Occasionally ligase will fail to join two cohesive ends and will instead leave a piece of linear DNA. This happens during the execution of our DNA Turing machine if a transition oligonucleotide is not incoporated in step 2, or one of the circularizations in step 4 or 6 fails. The "defective" linear tapes which result may be "chewed up" with a DNA exonuclease like Exonuclease III. DNA Exonuclease III only catalyzes the removal of DNA from an open $3'$ end so tapes that have closed correctly will be resistant to cleavage. This also means that if ligase manages to make one of the two covalent bonds needed to seal annealed sticky ends in our tape then Exonuclease III can only digest the strand with the nick. Further treatment with a single-stranded nuclease such as S1 will be necessary to degrade the other covalently-closed strand of the damaged tape.

(B) *Incorrect ligations*

DNA ligase can, under certain conditions, ligate mismatched pairs of sticky ends [**Wia**]. These incorrect ligations can only occur in our DNA Turing machine during step 2 and may happen in one of two ways. Identical copies of an `invariant` sticky end may be ligated (creating a mismatch at one position) or an incorrect oligonucleotide transition may be joined to the end left by the `state` cutter enzyme (creating mismatches at at least two positions). Tapes with such mismatches can be detected using a single-stranded nuclease such as nuclease S1 [**Bu**] which cleaves double-stranded DNA at the single-stranded regions induced by mismatch, or chemical reagents such as hydroxylamine or osmium tetroxide that modify mismatches and make them susceptible to cleavage by piperidine [**Co**]. Once the defective tapes have been identified by linearization, subsequent treatment with Exonuclease III can remove them from the computation.

(C) *Failed restrictions*

Restriction digests are not always complete. After steps 1, 3, and 5 some of the DNA tapes may still carry the head (`Inv` and `Sta`), `Cap`, or `X` restriction sites, respectively. Before the computation can proceed these defective tapes must be removed from the reaction mixture to keep them from interfering with later steps.

Just as a biotin label incorporated into a `Halt` sequence may be used to remove DNA tapes that have finished computing, biotin[27] or other distinct reporter molecules might be used to mark and retrieve tapes from which the head, `Cap`, or `X` sites have failed to be cut. Extra nucleotides can be added at the end of the `Cap` sequence, between the back to back `invariant` and `state` recognition sites, or between the back to back occurrences of the `symbol-excision` recognition sites to carry the reporter groups. After each restriction all DNA tapes which still have the reporter (which should have been cleaved in the last step) may removed using affinity chromatography.

There is no shortage of distinct reporter molecules with which to mark the 3 different functional parts of a transition oligonucleotide. Cholesterol, fluorescein, and dinitrophenyl groups are all available for direct incorporation into oligonucleotides from *Clontech* [**Cl**]. The latter two may be retrieved with antibodies. Single-stranded oligonucleotides "side chains" may also be incorporated into the transition oligonucleotides with the use of branched phosphoramidites. Oligonucleotides complementary to these side chains would be used to remove tapes with failed ligations allowing a virtually infinite variety of reporter molecules.

(D) *Incorrect restrictions*

Errors in which a restriction enzyme cuts DNA at a site other than its recognition site are extremely infrequent and have not been well quantified for all restriction enzymes. When restriction enzymes do cut incorrectly it is under non-standard reaction conditions and generally occurs at sequences closely related to their recognition sequence [**NEB**]. This means it is good practice to run restriction reactions in their suggested buffers and to minimize the similarity between a particular restriction site and the DNA which surrounds it. Still, this error mode really would have to be explored experimentally under the reaction conditions used for the DNA Turing machine.

(E) *Dimerization of tapes during cyclizations*

[27] Suggested by Nadrian Seeman, personal communication.

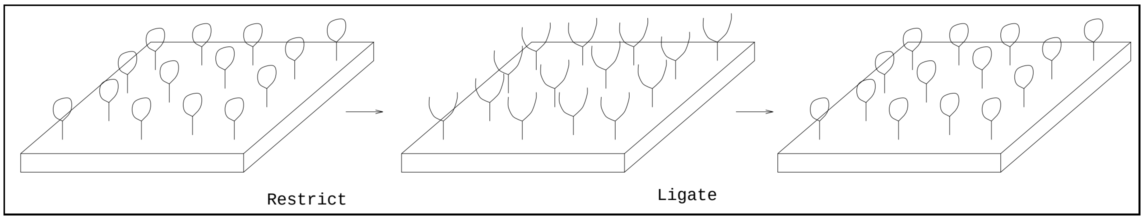

Figure 19: DNA restriction and ligation on solid support.

In (B) we describe errors in which ligase does the "wrong thing" and joins two unmatched sticky ends. Here ligase does the "right thing" and ligates matched ends, but we choose to interpret it as an error. We have already mentioned the primary means for keeping two different DNA tapes with complementary ends from joining during the recyclization steps 4 and 6—the reactions are run under very dilute conditions so that any molecule of DNA "sees" its own cohesive end more often than those of others [**Sa**]. For similar reasons shorter DNA tapes (as long as they are not too short and stiff) will also cyclize better than longer floppier DNA tapes [**Cr**]. One design consideration that must be made for shorter DNA tapes is to ensure that the number of helical turns in the DNA tape after steps 4 and 6 is integral [**Sh**]. This guarantees that the circularized DNA is unstrained and unsupercoiled. To further promote intramolecular closure of the DNA tapes we might use any of a number of DNA binding proteins like catabolite activator protein (CAP) that are known to bend DNA and promote cyclization [**Ka**].

7.2 Solid support

Many systems have been developed for the covalent attachment of DNA to solid support ([**Fa**], [**We**], [**Zh**]). The benefits of performing DNA chemistry on solid support are twofold. First, the cleanup of successive chemical steps is greatly simplified. Reagents may be applied and then simply washed way. The DNA need not be reprecipitated and run on a gel to separate it from the enzymes oligonucleotides used in the last step. Second, solid support can effect the separation of different DNA tapes by maintaining them at a low and constant concentration. This partially solves the problem of tape dimerization of Section 7.1, (E) without the need to dilute and reconcentrate the DNA tapes every ID. Tapes that contain errors are less likely to interact with other correct tapes so that their removal is not so urgent.

Zhang *et al.* have performed multiple restrictions and ligations on solid support to synthesize multiply branched DNA molecules. Our Turing machine implementation requires nothing so complex. A simple loop encoding the computation attached to the substrate via a single branched junction would suffice. Figure 19 shows the succesive restriction and ligation of "eyelets" of DNA on a solid support. Detection of `Halt` could be accomplished by washing flourescent or radiolabelled probes for the `Halt` sequence across the solid support. Retention of label would indicate that the computation was done and ready to be cleaved from the solid support.

7.3 Specifications

Very loose estimates for various "hardware specifications" of our DNA Turing machine are given below:

(A) *Size*

The amount of DNA required for a "useful" computation is ill-defined. Therefore we give only the most basic measures of size for our implementation of Minsky's Turing machine, based on the 16 base pair symbols used to represent its 4 symbol alphabet: atoms—528 atoms/bit; mass—5.28 kg/mol bits; length—2.7 nm/bit; volume—8.5 nm^3/bit.[28] These estimates do not take into account the size or mass of the water or solid support used to maintain this DNA. Restriction digests are normally performed with DNA

[28] Assuming DNA is a 2 nm wide [**Stry**] "cylinder".

concentrations around 1 μg DNA / 50 μL water. This means we require about 260 M^3 of water to maintain a mole of bits—or about 1/10th the volume of an olympic sized swimming pool.

How big could an individual DNA tape get? We might want to keep our Turing machines rather small to make the crucial cyclizations in steps 4 and 6 as likely as possible. If, however, we were willing to operate our Turing machines at very low DNA concentrations then much larger tapes could be used. *E. Coli* maintains a 4,700 kilobase circular genome. A DNA tape of this size would have space for 300,000 16 base pair symbols. If, as before, the cells hold members of a 4 symbol alphabet and there are 8 bits per byte this gives us an 80 kilobyte memory per tape.

(B) *Speed*

Restriction reactions require varying amounts of time based on the particular enzyme, temperature, reaction buffer and enzyme concentration used. To a first approximation, the more one spends, the more enzyme one can use and the faster the reaction will proceed.[29] Restriction enzymes, however, especially class IIS enzymes[30], are not inexpensive. Further, it is recommended that some enzymes, notably *Fok* I, not be used to *overdigest* DNA. That is, the enzymes shouldn't be used at concentrations much higher than those recommended by the supplier. Even if cost isn't an object this puts one limit on our ability to buy faster reaction times with greater enzyme concentration. We assume, then, that our slowest restrictions are made at "normal" restriction enzyme concentrations so that a complete digest is achieved within an hour. Ligations run at 20°C may be completed in 30 minutes [**NEB**]. Assuming that all of the reactions can be performed on solid support so that cleanup is minimal, the 3 restrictions and 3 ligations performed to move from one ID to the next will take on the order of 4.5 hours.

(C) *Energy*

We described, in Section 2.2, Charles Bennett's vision of dissipationless computers. Unfortunately, our machines are not reversible—logically or chemically. The small UTMs we propose to implement do not have reversible transitions and hence lose information. As for the chemistry, restrictions are free, or at least they do not cost us extra energy. The ligations, however, require that one high energy phosphate bond, from the hydrolysis of an ATP molecule to AMP and PP_i, be spent every time a nick is sealed in the phosphate backbone. Two ATPs then, are required to join every pair of sticky ends. Three ligations are performed to move from one ID to the next during which we must spend about 44 Kcal/mole DNA tapes.[31]

Since, in principle, all of the ligation and restriction reactions we perform would be reversible at equilibrium, why don't we take advantage of this and build a near dissipationless computer? For our computer to operate correctly we run it as far from equilibrium as possible! Ligase will perform the "reverse" of its normal operation and nick double-stranded DNA in the presence of AMP [**Kor**] but it doesn't do so site-specificly. At equilibrium ligase would cut the DNA tapes willy-nilly and our computation would be hopelessly scrambled. For similar reasons we wish to keep the restrictions from running backward—the restriction enzymes have no way of "knowing", for example, which symbols were last excised from which tapes. In a population of tapes at different stages of computation or performing different computations this means that symbols would be randomly reincorporated in to our DNA tapes.

Bennett's computer gets away with operating at equilibrium by performing its transitions as single, site-specific atomic steps; the state, symbol and head position change are all performed at once by a single enzyme. We split our transitions up into steps which, in reverse, are not site-specific and can be mixed and matched to form transition oligonucleotides that were not part of our original Turing machine.

7.4 Flexibility of the model

In Section 6 we demonstrated that Minsky's 4 x 7 Turing machine (and hence all 4 x 7 TMs) could be implemented using commercially availably restriction enzymes. The product of the number of states and symbols (28 in this case) is sometimes taken as a loose measure of the complexity of a Turing machine. Applying this measure to DNA Turing machines we ask: What is the largest state-symbol product that our

[29] Using enzymes in a stoichiometric fashion like this is enough to make any chemist shudder.

[30] *Eco*R I is cheap as enzymes go—it costs $50 for an amount that can digest 10 mg of DNA in an hour. The class IIS enzymes *Fok* I and *Bsr*D I cost 10 and 100 times as much, respectively.

[31] Assuming about 7.3 Kcal/mol ATP hydrolyzed [**Stry**].

model can achieve? The answer depends on an number of factors including the specificity and compatibility of enzymes and the number of mismatches we require in incorrectly paired overhangs. Assuming, as before, that we would like to have 2 base mismatches between incorrectly paired overhangs and assuming that we may use all of the class IIS restriction enzymes known in the literature we estimate that our model can simulate Turing machines with a state-symbol product of about 60. Ultimately the size of the Turing machines that can be built with our method will depend on the discovery of new enzymes and our ability to engineer new restriction enzymes with new recognition and cleavage sites. A series of recent papers ([**Kim1**], [**Kim2**], [**Lin**]) demonstrate that *Fok* I can be mutated to give novel cleavage specificities.

This encoding of Turing machines actually yields a model of computation slightly more powerful than a single tape Turing machine. During the course of a computation the transition table may be changed. This could allow a set of small transition tables to be applied to a DNA tape sequentially, each acting as a subroutine for a larger computation. Additionally, each transition in the transition table need not write a single symbol. The "result" region of each transition oligonucleotide can have an abitrary result string that holds many symbols—or none. This allows us to add cells to the tape if we need more space, or delete cells if this speeds up the computation.

If we choose to run our Turing machines in solution instead of on solid support (Section 7.2) we might decide to take advantage of the plasmid's natural circular boundary condition to implement a different simple model of computation, a TAG system. A TAG system, like a Turing machine uses a 1 dimensional tape but, instead of moving back an forth on a tape, it continuously chews up one end of the tape, and appends strings on the other. If we "circularize" this model of computation and connect both ends of the tape with a read/write head between them we get a model that looks very much like our DNA model, but it moves in only one direction. A TAG system normally reads a symbol, writes a string that is a function of that symbol and then erases some constant number of symbols, `P`. We can encode this with our DNA model by using `P + 1` states. If the head state `p` is one of the first `P` states of the machine, then the transition oligonucleotide instructs the head to write no symbol, change to state `p + 1` and move to the right. If the head is in state `p = P + 1` it writes a result string specific to the current symbol, changes the state `p` to 1 and moves to the right. Termination in a TAG system can function just as it does in Turing machine with the incorporation of a special `Halt` sequence into the tape.

7.5 Prospects for molecular computation

Even though our model can perform slightly more complex transitions than a "normal" Turing machine it doesn't seem like our model can ever be much faster than single tape Turing machines—the head still moves one cell at a time. The question remains: Can small Turing machines, compute anything useful in reasonable time? Unfortunately, the smallest Universal Turing machines known are *very* slow. Minsky's machine takes an amount of time exponential on the size of the tape of the machine it is simulating to get from one ID of that machine to another. Stål Aanderaa [**Aa**] calculates that his "fairly small" 10 x 6 UTM[32] requires 20,000 steps to advance the simulation of a 2 x 2 Turing by one ID when the simulated machine's tape holds only 6 symbols! Machines that use unary representations of other machines, it seems, are doomed to be slow.

While Universal machines might not be practical Turing machines to implement with DNA, it is possible that smaller special purpose machines which take only very small polynomial time on the size of their input tapes (hopefully linear time!) might perform a useful computation. The dominant DNA computation paradigm (after Adleman and Boneh *et al.*) is to use clever selection techniques to find answers a search space of all possible problems. By reducing the size of the input to these selection based computers we may be able to speed them up. Perhaps small Turing machines could be used as preprocessors for other DNA computers. The real challenge here is to find useful input languages for selection based computers which small Turing machines can generate in a few steps that standard synthetic techniques cannot.

8 CONCLUSION

In this paper we used the operation we call "Progress" and the concept of cutting frames provided by class IIS restriction endonucleases to propose one way of encoding a Turing machine with DNA chemistry.

[32] Aanderaa's machine is too big to be implemented with our encoding using only commercially available enzymes.

We assigned real DNA sequences to our schematic that could be used with commercially available enzymes to implement a Turing machine in lab. Finally, we recognized that the Turing machine model, while useful for proving theoretical results, may be too slow to do any practical computation. We do hope that our demonstration that one subset of DNA chemistry can field a Universal Turing machine will motivate others to study the chemistry of biology and come up with a series of operations from which one can build a *practical* universal molecular computer.

ACKNOWLEDGEMENTS

I would like to thank Max and Judith Rothemund for the generous grant that made my studies at the California Institute of Technology possible. Yaser Abu-Mostafa (dept of electrical engineering California of Technology) provided great enthusiasm, encouragement, and input for what was an unusual project. Erik Winfree (CNS Caltech) has kept me working and without him, I never would have attended DIMACS. I would like to thank Kai Zinn (dept of Biology California Institute of Technology) for letting me use DNA Strider early on; Greg Fu (dept of chemistry Massachusetts Institute of Technology) and Randy Lee (dept of Chemistry University of Houston) for encouraging my interest in chemistry; Sam Roweis and Len Adleman for their stimulating discussion; Nadrian Seeman for discussion on error correction; and Aditi Dhagat, Barun Chandra, and Amitabh Shah for introducing me to CS.

Finally, I would like to dedicate this paper to the late J.L.A. van de Snepscheut, (Dept of Computer Science California Institute of Technology) for introducing me to the idea of a DNA Turing machine [**Va**]. He would have been excited and pleased by the development of DNA computation.

For complete simulations of the schematic DNA Turing machine presented in this paper, explicit sequences for the BB-3 transition oligonucleotides, or a simulation of the BB-3 machine's first timestep using these oligonucleotides, please browse http://www.ugcs.caltech.edu/~pwkr/oett.html or make a request by email to pwkr@alumni.caltech.edu.

References

[**Aa**] Aanderaa, S., Jervell: A Universal Turing Machine. *Computer Science Logic, Lecture Notes in Computer Science* (1993) 1–4.

[**Ab**] Abu-Mostafa, Y.: *Notes on Information and Complexity* (1992) sec. 2.2.

[**Ad1**] Adleman, L.: Molecular computation of solutions to combinatorial problems. *Science* 266 (1994) 1021–1024.

[**Ad2**] Adleman, L.M.: On Constructing a Molecular Computer. Draft, January 8, 1995.

[**Ben1**] Bennett, C.H.: Logical Reversibility of Computation. *IBM Journal of Research and Development* 17 (1973) 525–532.

[**Ben2**] Bennett, C.H.:The Thermodynamics of Computation—a Review. *International Journal of Theoretical Physics* 21 (1982) 905–940.

[**Bu**] Burdon, M.G., Lees, J.H.: Double-strand cleavage at a two-base deletion mismatch in a DNA heteroduplex by nuclease S1. *Bioscience Reports* 5 (1985) 627–632.

[**Bon**] Boneh, D., Dunworth, C., Lipton, R., Sgall, J.: On Computational Power of DNA. To appear.

[**Co**] Cotton, R.G.H.: Detection of single base changes in nucleic acids. (Review) *Biochemical Journal* 263 (1989) 1–10.

[**Cl**] *Clontech Catalog.* Clontech, Palo Alto, CA 1993/1994.

[**Cr**] Crothers, D.M., Drak, J., Kahn, J.D., Levene, S.D.: DNA Bending, Flexibility, and Helical Repeat by Cyclization Kinetics. *Methods in Enzymology* 212 (1992) 3–31.

[**Fa**] Fahy, E., Davis, G.R., DiMichele, L.J., Ghosh, S.S.: Design and synthesis of polyacrylamide-based oligonucleotide supports for use in nucleic acid diagnostics. *Nucleic Acids Research* 21 (1993) 1819–1826.

[**Ho**] Hopcroft, J.E., Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Pub. Co. (1979) sec. 7.6.

[**Hj**] Hjelmfelt, A., Weinberger, E.D., Ross, J.: Chemical implementation of neural networks and Turing Machines. *Proc. Natl. Acad. Sci.* 88 (1991) 10983–10987.

[**Lin**] Lin, L., Chandrasegaran, S: Alteration of the cleavage distance of *Fok I* restriction endonuclease by insertion mutagenesis. *Proc. Natl. Acad. Sci.* 90 (1993) 2765–2768.

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

[**Ka**] Kahn, J.D., Crothers, D.M.: Protein-induced bending and DNA cyclization. *Proc. Natl. Acad. Sci.* 89 (1992) 6343-6347.

[**Kim1**] Kim, Y-G., Chandrasegaran, S.: Chimeric restriction endonuclease. *Proc. Natl. Acad. Sci.* 91 (1994) 883–887.

[**Kim2**] Kim, Y-G., Lin, L., Chandrasegaran, S.: Insertion and Deletion Mutants of *Fok I* Restriction Endonuclease. *Journal of Biological Chemistry* 50 (1994) 31978–31982.

[**Kor**] Kornberg,A., Baker, T.A.: *DNA Replication, 2nd Edition.* W.H. Freeman and Company, New York (1993).

[**Mi**] Minsky, M.L.: Size and Structure of universal Turing machines using tag systems., *Proc. 5th Symp. in Appl. Math.* American Mathematicl Society, Providence, RI, (1962) 229–238.

[**Nai**] Naito, T., Kusano, K., Kobayashi, I.: Selfish behaviour of restriction-modification systems. *Science* 267 (1995) 897–899.

[**Nas**] Nassal, M.: Total Chemical synthesis of a gene for hepatitis B virus core protein and its functional characterization. *Gene* 66 (1988) 279–294.

[**NEB**] *New Enland Biolabs Catalog.* New England Biolabs, Beverly MA 1995.

[**Sa**] Sambrook, K.J., Fritsch, E.F., Maniatis, T.: *Strategies for Cloning in Plasmid Vectors in Molecular Cloning Lab Manual.* Cold Spring Harbor Press (1989).

[**Si**] Siegelmann, H.T.: On the Computational Power of Neural Nets. *Journal of Computer and System Sciences* 50 (1995) 132–150.

[**Sh**] Shore, D., Baldwin, R.L.: Energetics of DNA Twisting. *Journal of Molecular Biology* 170 (1983) 957–981.

[**Stry**] Stryer, L.: *Biochemistry, 3rd Edition.* W.H. Freeman and Co. New York (1988).

[**Sz1**] Szybalski, W., Kim, C.K., Hasan, N., Podhajska, A.J.: Class-IIS restriction enzymes - a review. *Gene* 100 (1991) 13–26.

[**Sz2**] Syzbalski, W.: Universal restriction endonucleases: designing novel cleavage specificities by combining adapter oligodoxyneucleotide and enzyme moieties. *Gene* 40 (1985) 169–173.

[**Tu**] Turing, A.: On computable numbers with an application to the Entscheidungsproblem, *Proc. Math. Soc., series 2* (1936).

[**We**] Weiner, M.P., Felts, K.A., Simcox, T.G., Braman, J.C.: A method for the site-directed mono- and multi- mutagenesis of double-stranded DNA. *Gene* 126 (1993) 35–41.

[**Wia**] Wiaderkiewicz, R., Ruiz-Carillo, A.: Mismatch and blunt to protruding-end joining by DNA ligases. *Nucleic Acid Research* 15 (1987) 7831–7848.

[**Va**] Van de Schnepscheut, J.L.A.: *What Computing is All About.* (1994) sec. 11.5.

[**Zh**] Zhang, Y., Seeman, N.C.: A Solid-Support methodology for the Construction of Geometrical Objects from DNA. *Journal of the American Chemical Society* 114 (1992) 2656–2663.