DNA computers in vitro and vivo

Warren D. Smith*
wds@research.nj.nec.com

date = March 20, 1995

Abstract — **We show how DNA molecules and standard lab techniques may be used to create a nondeterministic Turing machine. This is the first scheme that shows how to make a universal computer with DNA. We claim that both our scheme and previous ones will work, but they probably cannot be scaled up to be of practical computational importance.**

In vivo, many of the limitations on our and previous computers are much less severe or do not apply. Hence, lifeforms ought, at least in principle, to be capable of large Turing universal computations.

The second part of our paper is a loose collection of biological phenomena that look computational and mathematical models of computation that look biological. We observe that cells face some daunting computational problems, e.g. gene regulation, assembly of complex structures, and antibody synthesis. We then make simplified mathematical models of certain biochemical processes and investigate the computational power of these models. The view of "biology as a computer programming problem" that we espouse, may be useful for biologists. Thus our particular Turing machine construction bears a remarkable resemblance to recently discovered "RNA editing" processes. In fact it may be that the RNA editing machines in *T.Brucei* and other lifeforms are clonable, extractible and runnable in vitro, in which case one might get a far better performing Turing machine than all constructions so far, including our own. The fact that RNA editing *is* a Turing machine may in turn have a lot to do with the origins of life. We also have a computer science explanation for "junk DNA."

Keywords — DNA based computing, nondeterministic Turing machine, RNA editing, transposons, replicon killers, cellular automata, multicellular development, gene regulation, hypotrichous ciliates, junk DNA, biologically based models of computation, NP, PSPACE.

1 Introduction

We will show that DNA and standard biochemical lab techniques may be used to create molecular-size Turing machines in vitro. All our arguments are based on procedures and numbers from the extensively cited experimental literature but we emphasize that we have *not* carried out any of these procedures. Turing machines[1] are simple but *universal* computers; that is, you can write a program to run on a Turing machine which will emulate *any* computer performing *any* computational task, such that this emulation will have a slowdown ratio bounded by a *power law.* Because they are DNA molecules, our Turing machines are available in huge numbers, and the programmer can insert *nondeterministic choice* steps into his program which 50% of the molecules[2] will perform one way, the rest the other way. The result[3] is huge parallelism.

The plan of this paper is as follows:

1. We discuss some previous proposals for DNA-based computing.

2. We argue that none of them are competitive with available algorithms on conventional computers.

3. This is at least partially due to the fact that the previous proposals were not universal computers. The second section of this paper shows how to make one.

4. Although several other authors have now also devised universal computers based on DNA, our paper, in our opinion, gives the most biochemically realistic discussion. Our quantitative estimates show that the present scheme, and the other authors's schemes *still* look impractical.

5. Because of this, we think that computer scientists who want to work in this area might more productively devote themselves to going in the other direction – i.e. seeing what computer science can say about biology, rather than asking what biochemistry can contribute to computer engineering.

*NECI, 4 Independence way, Princeton NJ 08544

[1]For terms from computer science such as NP, PSPACE, RAM, SAT, Turing Machine (TM), and nondeterministic Turing machine (NDTM), see appendix A, Garey & Johnson 1979, Minsky 1967, or for a popular exposition, Dewdney 1989. For background in biochemistry, see the books in our bibliography, e.g by Lewin, Stryer and Alberts et al. and our tutorial appendix C.

[2]Actually, any desired percentage is possible. Such steps split the program into two programs operating in parallel from then on.

[3]Our NDTMs can be extended to have two additional capabilities not usually seen in previous discussions: substring insertion capability (cf. §2.2), and an optional and peculiar "random lossy broadcast" interprocessor communication mechanism (cf. footnote 28 in §2.5).

6. So in the third section of this paper, we survey biological phenomena that look computational combined with some speculation about computational models to explain some important biological phenomena. Among these: we have "computer science explanations" for "junk DNA" and "RNA editing in trypanosomes."

7. Last but hardly least, our notion that "RNA editing" is in fact a natural molecular general purpose computational device restores hope that "DNA computing" might be practical, because as we point out, it may be possible to reprogram RNA editing machinery to do whatever we desire. If this works, it will work at continuous speeds of around 1 operation per second, which is thousands of times faster than all the other proposals, including our own.

1.1 Previous work on DNA computers

The notion of using DNA molecules to do computing originated[4] in a paper by Adleman 1994. Adleman showed how to use DNA to solve the "Hamiltonian path problem in directed graphs."

He choose a very simple example, which could be solved by inspection: given a set of 6 "cities" with some interconnecting (directed) "roads," find a path which visits each city exactly once. Adleman synthesized unique short (20-mer) nucleotide "tag" sequences to give each city a "from" and "to" address. Each road was represented by the 40-mer containing the appropriate city tags in the correct order. In solution these single-stranded DNA molecules randomly *hybridized* to their complements, forming longer strands which were double-stranded in complementary regions. After allowing ≈ 4 hours for the strands to hybridize, he used DNA ligase and polymerase to "sew up" the strands, getting set of DNA molecules representing all possible paths in the graph. He then performed chemical and physical operations to extract only the DNA molecules which corresponded to *Hamiltonian* paths from this soup[5].

The total number of separation and synthesis steps required grows only linearly with the size (number of cities and roads) of the problem; but remember that an exponentially large number of DNA molecules are required, so this linear bound only works up to a certain point. Adleman's idea is that since the number of DNA molecules in a soup is rather large, we get a large constant speedup compared to a sequential implementation of the same brute force procedure.

Exactly how large this parallelism factor can be made is not clear – since Adleman only solved a 6 city problem – but, counting molecules, it is at least 10^{11} and no more than 10^{23}. Each individual DNA step took Adleman about a day, although improvement to under an hour is thinkable. Meanwhile the comparable steps on a modern digital computer take on the order of 10^{-8} seconds. Thus, the maximal speedup factor obtainable from these DNA methods is somewhere between 10^{-2} and 10^{+11}.

Lipton 1994 then showed, beautifully simply, how to use the same primitive DNA operations as Adleman to solve any "SAT" problem (Garey & Johnson 1979) with N binary inputs and G AND, OR, or NOT gates, in a number of operations depending linearly on $N+G$. Lipton estimated, optimistically in our opinion, that it would be feasible to solve such problems, using his procedure, as large as $N=70$.

1.2 Reality: DNA computers aren't competitive

We would now like to express some skepticism. First, ordinary sequential computers are not restricted to using naive brute force algorithms to solve NP-complete problems such as Hamiltonian path and SAT, although the Adleman and Lipton constructions are. Instead, they can use clever "branch and bound" algorithms, which, while still exponential, tend in practice (and[6] in some cases provably) to have much smaller growth constants than the naive algorithm.

The development (Crowder 1980, Padberg & Rinaldi 1987) of programs for rigorous solutions of traveling salesman problems had advanced by 1987 to the point where a 9000-line FORTRAN program on a Cyber 205 solved a real life fully-connected 2392 city TSP in a 27-hour run[7]. Adleman's DNA method, despite its large parallelism factor, couldn't do a 30-city TSP, since $30! \approx 4 \times 10^4 \times N_{\mathrm{Avogadro}}$. Even "hard" (Mitchell-Selman-Levesque 1992) random 3CNF SAT problems, with $N=300$, are solvable by conventional computers (Dubois et al. 1994, Selman et al. 1992) in an average time of 10-20 minutes. This lies rather beyond Lipton's optimistic estimate of $N=70$ for the capabilities of his DNA based method. The moral is that good algorithms can often buy you speedups much larger than a mere fac-

[4] Hjelmfelt et al. 1991 had earlier conceived of the possibility of a chemical Turing machine, but they did not propose any specific chemicals, nor was their abstract published construction complete.

[5] Adleman's algorithm can be extended. For example, to find the *shortest* Hamiltonian path in a graph with small integer edge lengths, use DNA fragments, corresponding to the directed edges, with appropriate lengths and do a final electrophoresis step to find the shortest DNAs. Incidentally, we should point out that there is room for skepticism that Adleman's experiment actually worked as advertised. The final DNA was not sequenced to exhibit the Hamilitonian path explicitly; it was merely concluded indirectly that that is the DNA that must have been produced. Also it seems possible that the initial step of creating "all possible paths" could in fact have been random annealing and random ligating, producing all possible sequences of edges and cities, whether paths or not. However, that would not in principle have affected the validity of Adleman's algorithm! – it would merely have increased the size of its exponential resource requirements.

[6] The naive algorithm for SAT takes 2^N steps. Monien and Speckenmeyer 1985 proved that their algorithm for 3-SAT runs in $O(1.62^N)$ steps. An algorithm by Crawford and Auton 1993 solves "hard" 3-CNF instances in time which empirically seems to grow like $2^{N/17}$.

[7] This record (for L_2 distance, points in the plane) is now up to 4461, and the solving time for the 2392-city problem is down to 2.6 hours. TSPs with other distance functions such as $\{1,\infty\}$ and random distances in $[0,1]$ seem to be much easier.

tor of 10^{12} speedup in a naive algorithm[8], so that Adleman and Lipton's DNA methods are not competitive.

Open problem #1: *Can anybody find a class of useful problems for which some DNA based computing scheme clearly will outperform conventional computers?*

Still, molecular computing may have the potential to improve upon the capabilities of conventional sequential electronic computers, and this potentiality is worth investigating.

1.3 Better algorithms wanted - need universal computer

The way in which the good algorithms manage to achieve such good results on SAT and TSP is, extremely crudely speaking, only to branch when they must. Thus when solving SAT, one might prove, by a backtrack search from the current configuration and the use of bounding theorems, that the next three bits must be either a '100' or '110.' In that case, we do not need to consider all 8 possibilities, but only 2 of them, and if this sort of thing were to happen every time, the cost of the search would be $2^{N/3}$ instead of 2^N. The programs that take advantage of such tricks may be thought of as programs on a *nondeterministic Turing machine* but in which the invocations of nondeterminism are carefully controlled and made as infrequent as possible.

We want to be able to duplicate this capability (of allowing clever programming, in which nondeterministic steps are done only at carefully selected places in the program) with DNA, so that we may have our cake (clever algorithms) and eat it too (molecular scale parallelism).

In the present paper, we show how to do this.

1.4 What we do

More precisely, we show that molecular biology lab techniques ought to be[9] useable to create a "nondeterministic Turing machine" (NDTM).

For problems in which memory requirements only are polynomially growing functions of the size of the input ("PSPACE"), Turing machines can indeed emulate any other computer with slowdown ratio only polynomial *in the size of the input.* Any program (with nondeterministic statements allowed) for a conventional computer may be executed on an NDTM, and if the (worst case, or average case) time required by the conventional computer is bounded by $P_1(N)A^N$ where N is the number of bits in the input, $A \geq 1$ is a constant, and $P_1(N)$ is a polynomial, then the NDTM will require resources bounded by $P_2(N)A^N$, where P_2 is another polynomial[10]. Thus any clever algorithms which tend to reduce the growth constant A, will still be that way in the DNA version[11]. Also, so long as we are below the point at which we run out of DNA, our NDTM will solve any problem in NP in polynomial time[12].

2 In vitro DNA computer

2.1 DNA NDTM: description and operation

Our Turing machine is a circular loop of partly single and partly double stranded DNA. Actually there are a huge number of such loops, which is essential to give us nondeterministic parallel capability, but let us concentrate on one of them for clarity. Initially, all the TMs are in identical starting states.

Most of the DNA sequence is divided into chunks representing the characters on the Turing machine's "tape." Suppose the chunks are 20 bases long and our Turing machine has an alphabet of size 26. Then we would need to designate 26 special 20-long sequences to represent the letters of the alphabet. There are certain constraints on the choice of these sequences: they need to be dissimilar and they need to avoid long palindromes[13] and certain

[8] Nor would the results of Lipton's or Adleman's DNA methods, if they were negative, be a rigorous nonexistence proof, though the probability of error could be made small. Neither Adleman nor Lipton analysed the errors in, nor the physical, engineeering, and rate and thermodynamic constraints on, their processes and how they would behave upon scaling up from toy problems to ones large enough to stymie conventional computers. Actually, this may not matter much, since their exponential growth factors are so bad that they *can't* be scaled up to become competitive. But since *we* have a general purpose computer, we care about scaleup – and the situation looks bad. We'll see that just the ambient rate of DNA damage alone will lead to very severe limitations.

[9] We again emphasize that we have not actually carried the experiment out, unlike Adleman.

[10] This applies for all the algorithms for TSP and SAT we have mentioned. However, EXPSPACE algorithms, e.g. running in time $O(2^N)$ and consuming 2^N bits of memory, so far as is currently known might require time $\approx 4^N$ to run on a Turing machine.

[11] Assuming they parallelize without much loss. Chakrabarti et al. 1994 and Karp-Zhang 1993 support the notion that backtrack search can be parallelized optimally with high probability. Although these authors are using a computational model which supports interprocessor communication, it turns out that the only sort of communication one needs is precisely the sort of "random broadcast" operation we are capable of supplying (cf. footnote 28), and the "lossiness" there can be overcome, with high probability, by repeated broadcasts.

[12] Some other authors claim to have gone beyond our construction. Our construction shows, crudely speaking, how to solve problems in the class NP with a DNA computer in "polynomial" time. Both Don Beaver (1995) and later John Reif, claimed to have shown how to use a DNA computer to solve problems (again speaking crudely) in the presumably larger class PSPACE. But in fact, Beaver's scheme involves DNA hybridization steps which will require exponentially long annealing times. Reif's scheme avoids that defect, but at the cost of square-rooting the amount of available parallelism; thus, the real class of problems that Reif can solve might better be colloquially described as "$\sqrt{\text{PSPACE}}$." Thus Reif's result does not dominate ours; it is in fact incomparable with it. (The fundamental problem with both schemes is the need to do DNA hybridization steps in soups in which there are exponentially many different kinds of DNA, each one of which is supposed to find its mate.) With the speeds and molecular numbers currently thinkable, this square rooting would cause Reif's scheme to run more slowly than conventional computers.

[13] Palindromes, by which we mean a substring S followed after some gap by its reversed complement, can cause hairpins which can stall polymerases and cause bacterial mutation hotspots, (Challberg & Englund 1993). We also wish to avoid translated copies of short substrings to prevent unintentional DNA pairings.

The "modified (k, A) DeBruijn sequences" of appendix B with $A = 4$ are a convenient source of DNA strings which completely avoid mismatches $\geq k$ base pairs long and also completely avoid restriction enzyme subsequences; but these constraints seem to be satisfiable, see appendix B.

Between each such chunk is a separating "comma," which is another special 20-long sequence.

The tape is thus not infinite, but in fact is finite and circular. Since we seriously doubt that anybody will run our NDTM for more than 10^5 steps (and also since, as we will later see, our Turing machine can choose to create more tape) this is not a limitation.

At the far side of the circle (from the head), one would place a special "anchor" subsequence, which could serve to delineate "ends" of the tape and would more importantly be used for anchoring each DNA at a fixed location, rather than allowing it to float freely around in the soup. This anchoring may be accomplished by the incorporation of special "biotinylized bases[14]" into DNA in place of the usual nucleotides by the use of nonspecific polymerases; and then surfaces coated with streptavidin will strongly bind the biotinylized bases[15].

Finally, there is a subinterval of the DNA that is going to represent the single "head" `H` of the TM and the character `L` immediately to the left of it.

The scenario is depicted in figure 1. Here "`d,o,g,`" are normal characters on the tape, as are "`c,a,t,...`" but `L`, `H`, and `>` are special and pertain to the head `H` and its immediately adjacent tape square `L`. Actually `L` is also a normal tape character; the only way in which it is special is the fact that it lies immediately to the left of the head. Note, our Turing machine's head is located *between* two adjacent tape squares rather than on top of one of them. It decides what to do next based on the current head state (here `H`) and the character (here `L`) to its left.

2.1.1 7-step process to perform a TM transition

The **first** step is to use a restriction enzyme to cut the DNA at the right of the special subsequence indicated by the symbol '`>`'. This converts our DNA from circular to linear and leaves "sticky ends." We then remove the restriction enzyme from the container. Since the DNAs are anchored in place, this is simply managed by washing it. At the same time we "melt" the DNA to convert[16] it to single stranded form. (By using hot water, 94-97°C, i.e. well above the melting temperature, which for *E.Coli* DNA is 69°C and for *P.aeruginosa* is 77°C. The non-anchored strands are washed away.)

Restriction enzymes are enzymes which cut double-stranded DNA, often leaving "sticky ends" because they cut the strands at different locations, at specific recognition subsequences. Over 1500 that are known (Roberts 1978, Brown 1991 app. 3) and a few examples are below. Of these, only *Eco*RII and *Bbv*II would at first seem suitable for playing the role of "`>`," since only they have cut sites at the end of the recognized region. But actually, if we were to adopt the convention that, e.g., every character on the tape begins with A, then one could use *Hin*dIII, and so forth. (Here A=adenosine, T=thymine, G=guanine, C=cytosine, W={A or T}, N={A,T,G, or C}, and the bottom strand is here written in the $5' \rightarrow 3'$ order in which polymerases synthesize DNA. Of the below, only *Hin*dIII, *Hpa*I, and *Sfi*I were commercially available in 1991.)

```
enzyme name    recognized sequence
               and cut | locations
                     GGWCC|
EcoRII              |CCWGG

                GGANN N|NNTCC
EcoNI           CCTNN|N NNAGG

                CTTCTGNN NNNN|
BbvII           GAAGACNN|NNNN

                T TCGA|A
HindIII         A|AGCT T

                CAA|TTG
HpaI            GTT|AAC

                CC GC|GG
NarI            GG|CG CC

               CCGG|NNNN NCCGG
SfiI           GGCC NNNN|NGGCC
```

k-long palindromes. The previous works by Adleman and Lipton could make convenient use of them too. It is not clear how large one can allow k to be before mismatch and palindrome problems become dangerous (annealed DNA is known to mispair, Britten & Kohne 1968; the melting temperature decreases by ≈ 1°C per 1% mispairing, according to Wetmur 1976 and Bonner et al. 1973), in either our or the previous works. Even if $k = 5$ is the upper limit, we could still generate $25 \approx 516/20$ tape symbols, which is more than sufficient for Turing completeness (Minsky describes a universal TM with 4 symbols and 7 states; UTMs are now known with smaller (symbol,state) product than the value of 28 for Minsky's machine, for example, Yu. V. Rogozhin found a (6,4) machine, see Robinson 1991.), although not capacious elbow room. Meanwhile, this limitation could put Adleman and Lipton in severe trouble. If Lipton wants to avoid unhappy marriages of length 5, then all C of his L-long DNA chunks must not contain any 5-long substring twice, so $C \cdot (L-4) \leq 4^5 = 1024$, so with $L = 20$, Lipton would be forced to have $C \leq 64$ and thus he could not solve SAT problems with more than 64 inputs; and the situation is twice as bad if one is also worried about palindromes.

[14]Discussed in appendix C

[15]See Green 1975, Ruth 1991, and *Methods in Enzymology* volume 184 for reviews of biotin and avidin techniques and properties. (Basics in appendix C.) The fact that the biotin-streptavidin complex, with a heat of formation of 23 Kcal/mole, is very stable to heat (up to 132°C; it also is stable to pH 2-13 and proteases) will be important in our later discussion. One particularly impressive recent technique (Fodor 1993) involving photosensitive endcaps allows one to anchor DNAs to a surface and then add to one end of each bound DNA molecule, a sequence specifying its xy coordinates on the surface, as a binary number!

[16]In fact, fully circular DNA cannot melt, since the two strands are linked. When we say "circular double stranded" DNA we really mean that the second strand does not continue all the way around; the first strand is actually bare in the region near the anchor and only double stranded in the region near the head. This could be assured by the use of bulky artificial biotinylized bases which simply won't pair. See also §2.4.

The time needed for this step should be between 10 and 100 minutes (Sambrook 5.31, Halford & Johnson 1980).

The *Eco*RI restriction enzyme cuts at its recognition site (1-3) $\times 10^5$ times more quickly than it makes an erroneous cut at a random site according to Terry et al. 1983, while Halford & Johnson 1980 claimed 5×10^7. Corresponding cut rates on single strand DNA[17] are not known to us. Such imperfect specificity may set upper limits on the size of our Turing machine.

Second (after cooling it down), we pour in a mixture of single stranded DNA from a large container labeled "the transition matrix." This particular mixture is supposed to be available in large quantities and has a specific composition which is independent of the initial state of our Turing machine.

This transition matrix mixture contains some DNA strands which begin with the antisense of `, L , H >`, continues with a comma `,`, and further with the antisense of the *new* `L* H* >`, subsequence which would result from the TM performing one step, e.g. overwriting `L` with a new character `L*` and changing the head state to `H*`, and ends with the antisense of `,`. In the event the TM step called for was *not* overwriting `L`, but instead moving the head left, the middle part of the transition matrix string would be of the form `, H*> , L*`, instead of `, L , H >`.

In the event that a nondeterministic combination of these two operations was called for, then both these strings would be present in the transition matrix. In fact, the "transition matrix" is the collection of all such strings for all possible transitions between all possible old `L` and `H`'s to new ones, that are allowed by the rules of the Turing machine. However, only the strings which pertain to the particular `L` and `H` in our particular TM will bind to it.

This binding will be *both* to the end of the single strand with the old `L` and `H`, and also to the other end of the same strand on the other side of the circle with a `,`. This makes the DNA circular once again, after a healing of the nicks by the application of ATP[18] and Ligase. For DNA the size of λ phage (48514 base pairs; 16μm) with sticky ends produced by a restriction enzyme, the circularizing procedure with ligase recommended in the PROMEGA catalog requires 20 minutes (in vivo, λ phage circularizes much more quickly than this); Shore et al. 1981 report that "all linear DNAs tested ranging in size from 242-4361 bp could be completely converted to covalently closed circles in < 1 min at 20°C by sufficient amounts of T4 ligase." The annealing time required (for the transmission matrix strings to bind with their antisense sequences) is harder to estimate, but based on descriptions of similar procedures, kinetics estimates in Britten & Kohne 1968, Wetmur 1968, and Weiss 1968, and especially considering the comparatively short length of these sequences[19], less than an hour should be required.

Because our DNAs are physically separated because they are bound to anchors attached to a rigid object, this relinking to circular form will not accidentally chain several TM DNAs together! Also, note that multiple heads cannot append because the transmission matrix strings are unidirectional – they are not available in reversed order ($3'$ to $5'$) form!

Third, we use a polymerase and deoxy-nucleotide triphosphates[20] to "fill in the missing pieces" and regain double stranded circular form. (The reader may follow these steps on the figure.)

Stryer page 581 says E.Coli polymerases I,II,III have speeds of 10, 0.5, 150 bases per second. Higher speeds are available by resorting to the *Taq* polymerase commonly used in PCR[21] but at the cost of a higher error rate. But the replication machinery *in vivo* in E.Coli operates at a much faster rate than that, since generation time is 20 minutes and the DNA is 4×10^6 base pairs for a rate (taking bidirectionality into account, and assuming at most half of the generation time is actually spent replicating) of at least 3333/sec.

The in vitro error rate of the *Taq* polymerase commonly used in PCR is 1/9000 (Kunkel 1989) whereas the Klenow fragment of *E.Coli* gets 1/12000 in vitro. (Nonenzymatic DNA synthesis machines have error and early truncation rates of 1/200 per base, cf. appendix C.) Again, the situation in vivo is much better. Stryer page 635 says that the error rate for T4 phage replication is 1.7×10^{-8} per base per replication. The same figures for *E.Coli* and *Drosophila Melanogaster* (fruit fly) are 4×10^{-10} and 7×10^{-11} respectively[22]. (Nonsubstitution errors are much rarer, unless one intentionally introduces certain intercalating mutagens. Also versions of E.Coli with defective polymerases exist, with much higher mutation rates.)

[17] We will use the fact that restriction enzymes only (except for unintentionally!) cut double strand DNA, later.

[18] ATP=Adenosine TriPhosphate, the molecule that serves as the fundamental "currency" of energy in the cell.

[19] Annealing times should grow proportionally to the square root of the effective length L of the DNA strands being annealed and to the reciprocal of the molar concentration: $t_{1/2}^{\text{anneal}} \approx \tau n \sqrt{L}/C$ for L bp long totally unrelated DNA fragments coming in n equinumerous species, where $\tau = 8.9 \times 10^{-6}$mole-sec/liter, and C is the molar concentration of nucleotides. For $C = 100\mu$M and $L = 100$, $n = 100$, this suggests $t_{1/2} < 100$sec. But this could lead to enormous annealing times in at least some instances of scaleups of the schemes by Lipton and by Adleman, e.g. if $n \approx 10^{17}$, $t_{1/2} \approx$the age of the universe...

[20] There is actually no need to "prime" the polymerase for the particular operation required here since all that is *really* needed is to "fill in the gap" on the upper strand in the figure; the lower strand does not really need to be extended as shown. In fact by using $3'$-dehydroxylated DNA strands in the transition matrix, if desired, such extensions could be prevented. We should also say that a ligase will be required to seal the nicks left by the polymerase.

[21] PCR=Polymerase chain reaction, an important technique for amplifying DNA.

[22] Drake 1991 proposed to explain such numbers, that all DNA based prokaryotes and viruses have an "optimized mutation rate" of ≈ 0.0033 mutations per genome per replication. RNA viruses (Drake 1993) have much higher mutation rates of order 1 per genome per offspring. The DNA replication enzymes alone have much higher error rates than these – of order 10^{-5} per base, and the increase in fidelity above this in vivo is due to additional checking and correcting processes.

Fourth, we melt the DNA with a flow of hot water to get back to single stranded circular form.

Fifth, we pour on some more transition matrix strings, except that these strings are slightly different than the ones from step 2 in that they do not extend to the `,` cut point after the new `H*` or `L*`. (In the figure we have called this the "truncated transition matrix.") After annealing we are left with single strand DNA loops but with the portions covered by the old `, L , H` up to the new `H*` or `L*` covered by a second strand.

Sixth, we expose these to a restriction enzyme that cuts at a point well inside the `,` sequence. Since restriction enzymes only cut double stranded DNA, this only cuts at the commas surrounding the old `L` and `H`, which is washed away, and leaves the rest of the DNA unaffected. The "sticky ends" left by the restriction enyme cut are then free to reattach, getting us back to circular but with the old `L` and `H` chopped out. Note that in the original restriction in the first step, no such reattachment (which would there, have been undesirable) was possible due to the immediate melting removing the second strand. Also the present reattachment is made permanent by using ligase to seal the nicks.

Seventh, we use polymerase to get back to double stranded form[23], and we have now completed the TM transition and are ready to start again at step one.

By iterating this 7-step procedure, one may do as many TM transitions as desired. Note that any incomplete reactions will generally either result in a 'no op' or else be corrected/completed next 7-step cycle, so our procedure is fairly robust.

Caveat: For simplicity, we have described a "1-way" Turing machine in which the head can only stay still or move left, and never move right. In fact, it is easy to add some extra bells and whistles of the exact same sort (using different restriction enzymes), to permit the head to move right. We have in fact worked out the details of this, and an 11-step cycle is needed, not a 7-step one.

2.2 *Optional extra capabilities – input, output, substring insertion, interprocessor communication*

We can make our Turing machine have the capability, not normally seen in theoretical discussions of Turing machines, not only of overwriting a character on the tape, but in fact of *inserting* a substring into the tape, next to the head, in one step.

This leads to interesting possibilities concerning input and output. Suppose the initial tape is totally blank (a periodic sequence blank, comma, blank, comma, actually) except for a head. The head will contain a restriction site of an additional flavor, so that by an single initial insertion of a giant substring (this could be done in vitro with the use of "integrase" and "IHF" the same way λ phage inserts itself into the host genome, or more conventionally with restriction enzyme cuts and ligase) we could write our {input and program} string to all the Turing machines in one initial step. That would take care of input.

The reader may wonder how the initial blank tape is synthesized. It is feasible to synthesize long *periodic* (blank comma blank comma...) DNA with period < 100 base pairs (see Khorana 1968 for the early work in this direction) by synthesizing the periods and linking them with ligases, even though it is infeasible to synthesize a comparably long arbitrary DNA sequence. More importantly, the reader may wonder how the initial program and data tape is synthesized, considering that currently available DNA synthesis machines can only make DNA about 100 base pairs long A slick answer to to both these synthesis problems is to use the Turing machines themselves to synthesize their own input! Specifically, the Turing machine begins by invoking a simple "bootstrap loader" whose purpose is to insert the tape characters forming the input and program (and a handy hoard of blank tape) one (or a few) at a time, using the insertion capability we mentioned earlier. Error correction will be discussed next section.

As for output, in many theoretical discussions of Turing machines it suffices merely to be able to tell whether the machine has entered the "halted" state. For that purpose we can expose our strands to antisense DNA (antisense of 'successfully halted head'), bound to dye. The successfully halted TMs will[24] then be colored. But sometimes one wishes to examine the whole tape. For this purpose one can release all the DNA anchors (photosensitive biotin anchor systems are available which break on exposure to light) and isolate successfully halted TM DNA using the same methods (based on paramagnetic beads bound to antisense DNA for the "halt" state) as were used by Adleman. After amplifying this DNA, e.g. inside bacteria, (by isolating a single bacterium and growing it into a colony, one gets a large number of reliable copies of one particular DNA molecule) one may then sequence it.

2.3 *Speed, error rates*

For Turing machines with tapes the rough size of λ phage (48514 base pairs), adding up our estimates shows that the 7 step cycle above, which implements a single Turing machine transition, will take a few hours, and will involve an error rate (assuming the steps involving polymerase incur the lion's share of the error) of around 1 incorrect base per 5000 per cycle.

There is reason to believe that if we were to scale up the Turing machine to have a tape N times larger, then the time requirement per cycle would increase eventually proportionally to N^p, $1 < p < 1.5$ (with most of the time being required for the circularization steps, see Shore et al. 1981). Also, we would have to use a smaller amount of DNA (more widely spaced apart to prevent chaining

[23] Actually, it is probably better simply to anneal with the antisense to `> ,` to convert to double stranded form only in one small area.

[24] Another idea: Since we can choose the halt state, we could decide to make it the binding site for some known DNA-binding protein such as the *E.Coli* Lac repressor.

in steps 2 and 6) dropping like some inverse power of N. From the standpoint of theory, any such "polynomial slowdown" does not affect the theorems which say that a Turing machine can simulate any other computer with at most polynomial slowdown. But as far as practice is concerned, this is certainly a limitation.

It should be possible to reduce the error rate hugely. In the future better polymerase systems, e.g. with error checking and correcting, may become available for in vitro use. E.g. see Fersht & Knill-Jones 1983. By use of careful procedures and a polymerase from *Pyrococcus furiosus*, Lundberg, Shoemaker et al. 1991 showed that the error rate in PCR could be reduced to 1 nucleotide in 625000, confirmed by Brail et al. 1993.

For the present, we do not need to replicate the entire DNA strand each application of polymerase, it will suffice to terminate the reaction after only the region fairly close to the head `H` has been converted to double stranded form. Not only will this speed things up, it will also keep the total number of errors low, since errors in replication will only occur in a region within (say) a few hundred bases away from the head. Next, we could use error correcting codes in the transition matrix. We encode tape characters using a radix-4 block error correcting code, and the transition matrix will include corrections for noncodes, so that whenever the head passes over a tape character with a small enough number of substitution errors, it will correct it. As one example of such a code, we mention the quaternary "octacode" (Calderbank et al. 1994) which is a set of 256 words of 8 letters over an alphabet of size 4, such that any two codewords are ≥ 3 mutations apart. To be precise, this code is the linear combinations mod 4 of the rows of

```
11111111
01001213
00103323
00012311.
```

This code will suffice to correct single errors and will allow any choice of 4 bases in any particular 4 places. If we use A=0, C=1, T=2, G=3, then this code also has the property that any two codewords are ≥ 6 *weighted* mutations apart, where mutations between letters only 1 apart mod 4, which are comparatively likely (Stryer p635-7), have weight 1, whereas the remaining mutations $A \leftrightarrow T$ and $G \leftrightarrow C$ have weight 2. Thus it also provides error correction of all weight 2, and detection of weight 3, mutations at no extra charge.

It would also be possible to combine error correcting codes with known DNA-repair enzymes to repair errors throughout our single strand DNA. The method would be to pour in methylated antisense DNA of the tape characters (as before, encoded with radix-4 block error correcting codes) and use ligase to get to double stranded form. Then repair enzymes (Lahue 1989) which take the methylated strands as gospel will snip out erroneous bases on the original single strand and replace them with the right ones, and then a melt will leave us with a corrected single strand once again. Such repair cycles could be undertaken occasionally to repair the entire tape. This scheme actually may be better than the repair methods used by living creatures, and in principle could reduce the substitution error rate near to thermodynamic limits.

But even without any of these repair procedures (which are new, and not standard lab procedures at the moment, hence we are hesitant to rely on them), it should be clear that our methods will suffice to create a Turing machine which will work *for sufficiently small scale problems.* The difficulties all arise when the problems are writ large.

2.4 DNA hydrolysis dissolves your computer & other damage

Speaking of which, another, extremely serious, scaleup problem is that single stranded DNA is less stable to hydrolysis than double strand DNA – and quite a large fraction of the time, we have got single strand DNA sitting around in buffer. Hydrolysis *nicks* in double strand DNA are repairable with ligase, but such nicks in *single strand* DNA are not nicks, but cuts, and there is no way to repair the destruction. Thus, during a long Turing machine run, one may find[25] one's DNA computer dissolving! Our Turing machines could in principle be programmed to try to correct for all sorts of chemical errors occuring locally near the head; but randomly snipping the tape and then washing the head down the drain, is too much!

In practice such hydrolysis problems are usually caused by nucleases introduced by biological contamination and not by the rate of DNA hydrolysis in pure water[26]. According to Lindahl 1993, each human genome (3×10^9 base pairs) in each human cell loses about 2000-10000 purine bases (A & G) to thermal hydrolysis of their N-glycosyl linkages to deoxyribose (depurination), and also about 100 C's spontaneously convert to uracil per day. These rates would be respectively 4 and 150 times higher in single strand DNA. Saul & Ames 1986 also estimate that the human genome suffers 10^5 single strand breaks ("nicks") per day (and tabulate various other damage rate estimates). Fraga et al. 1990 also estimate 10^5 oxidative hits on DNA per day.

In other words, in single strand DNA at 37°C at pH 7.4 we may expect about 10^{-7} errors per base pair per hour plus 1.4×10^{-6} single strand cuts per base per hour.

[25] Incidentally, the same error rate, hydrolysis and scaleup problems arise in the schemes by Adleman and Lipton, the difference is that we've discussed them. A related problem is that fluid shear commensurate with the pressure of a thumb squirting DNA through a syringe, will tend to cut it into fragments of length $\leq$ 15Kbp (Kornberg p21). DNA in vivo is protected from this by supercoiling and spooling around histones.

[26] Even if one is extremely careful to avoid contamination from bacteria, minute skin fragments, etc, the enzymes one gets from the supply company are not exactly 100% pure and will contain traces of other proteins, including undesirable ones such as unknown nucleases.

The long term stability of genetic information in vivo depends on DNA repair enzymes. Since the situation in vitro is probably worse, these may be taken as probable *lower* bounds on the rate of destruction. Shooter & Merrifield 1978 observed that single strand T7 phage DNA was hydrolyzed in a bath of 0.3 molar NaOH and 0.7 molar NaCl at a rate of 1.6×10^{-7} breaks per base per hour at 20°C, and 10 times that rate at 37°C, while Freifelder & Dewitt 1977 found that single strand λ phage DNA rates of (1-4) $\times 10^{-6}$ breaks per base per hour in various pH media at 75°C. These rates, which arise from intentionally severe conditions, probably may be taken as *upper* bounds (...except when they are smaller than the lower bounds).

Open problem #2: *Can methods be found to reduce the magnitude of this hydrolysis problem?*

Freifelder et al. mentioned that P.F. Davison in unpublished work in $\approx$ 1962 found that DNA in sealed vessels in an N_2 atmosphere had an "extremely low hydrolysis rate," but this "was never followed up." Another possibility – emulating the method used inside our own cells – would be to try to create an almost entirely double stranded DNA computer that somehow uses nickases and/or single strand binding proteins to catalyse the state transitions. However, once one starts building up all sorts of machinery to protect, package, and repair DNA, one soon realizes that what one is trying to create, is also known as life.

For the present, the hydrolysis problem is severe enough to prevent scaleup to computational problems with large (or even moderate!) memory requirements.

2.5 Is this useful? No.

Despite the standard theoretical claims that Turing machines are only polynomially slower than anything else, that polynomial slowdown will probably still be enough to render the scheme suggested here impractical in comparison with the usual sort of computers. When you combine the slow speed with the figures above on the background DNA hydrolysis rate and the limits on restriction enzyme specificities, matters look even worse. (And we have already argued that Adleman's and Lipton's schemes are definitely not of practical interest as computers.) Let's examine this in more detail.

The operations we require for a Turing machine step form a simple 7-step cycle whose repetition could be automated. It seems plausible to us that (we are allowing the possibility of hypothetical future improvements improving the lower bound on the time interval) this cycle could be completed in somewhere between 1 and 200 minutes. With a parallelism factor of 10^{11} to 10^{23}, that would mean that our speedups, as compared to a 100MHz electronic implementation *of a Turing machine*, would be in the range 10^{-1}-10^{+13}. So far, so good, but the speedup ratio compared to a *sensible* electronic computer, on problems for which Turing machines are not especially well suited[27] (i.e., most of them), would be a lot less. Due to the large uncertainties in these numbers (!) we cannot be 100% sure, but we suspect that for most computations, little or no advantage will be obtainable.

Bottom line – a Revealing Calculation: Assume $10\mu g$ of 1000-bp loop DNA, i.e. 10^{13} molecules, each $.3\mu m$ in circumference, bound to some large (sponge-like?) surface, and that each 7-step cycle takes 1hr. Then 1000 NDTM steps on tapes with ≈ 30 characters could be accomplished in about 1 month, after which time computations would have to cease since a good fraction of the DNAs would have hydrolyzed. Meanwhile a 100MHz electronic TM would only have accomplished 2.5×10^{14} steps, i.e. 40 times fewer steps, *but* we doubt this factor of 40 is enough to overcome the handicaps of being a small Turing machine, and if you'd really cared you might have built 40 electronic TMs. □

However, we *may* be able to use a very large transition matrix, much larger than the matrices found in most theoretical discussions of Turing machines. This might make the task of programming the machine less onerous and involving less steps to do something interesting, than anyone (unhappily) familiar with Turing machines, might at first think. It would also allow the use of error correcting codes as discussed before. (Transition matrices with N times more elements will require N times more annealing time.)

One could also return Turing machines which have reached an unsuccessful halt state to the computational pool, by the crude method of programming such machines to back up along the nondeterministic computational tree, until they are ready to resume computing (in other words, not all backtracking has to be done using nondeterminism). Less crude methods are unavailable to us since we lack good interprocessor communication mechanisms[28].

Open problem #3: *What is really wanted is not a nondeterministic Turing machine, but rather, a nondeterministic RAM or PRAM. Can anybody show how to construct one with DNA?*[29]

Incidentally, although we argued early that "the maximal speedup factor obtainable from these DNA methods is somewhere between 10^{-2} and 10^{+13}" because the maximal parallelism is between 10^{11} and 10^{23}, we point out that methods might be conceived in which computations happen at more than one place on *each* DNA molecule, e.g. a multihead Turing machine or 1D cel-

[27] As a rare example of a task to which TMs are especially suited, consider the task of computing (or trying to) the fifth "busy beaver number" (Brady 1983, Dewdney 1985, Buro 1990, Michel 1993).

[28] Actually by the use of λ phage Integrase and IHF (integration host factor) enzymes, and/or the excision factor Xis (Ptashne 1992), one would be able to snip out and circularize substrings from DNA, and later reinsert them at in the DNA of other (random!) Turing machines. Integrase always does this insertion and excision at a specific 15 base pair sequence on the host (and message) DNA (Mathews & van Holde page 883). This would allow a rather strange "lossy random broadcast" communication mechanism.

[29] Without needing to resort to "biosteps" which actually require exponential time, and while still having a very large usable memory?

lular automaton. If that were managed, an additional factor of perhaps 10^2-10^6 would be exploitable. We do not know how to accomplish this using standard lab techniques, but certainly this would be relevant for modeling the computational capabilities of DNA in vivo.

2.6 Summary – our machine and previous ones and their problems

Many previous schemes live in an imaginary world in which all reactions have 100% yield, DNA never gets damaged, and all steps happen quickly, etc. For the first time in the present paper we have analysed the reality of the situation. Although our scheme is theoretically more powerful than some previous schemes which were not universal computers, it also does not suffer from some severe practical problems that afflicted earlier schemes. Despite this, our scheme *still* does not appear practical.

1. Ours is the first scheme that shows how to make a *universal computer* with DNA.
2. We also suspect that our scheme ought to be less prone to error and simpler to do than the previous schemes of Adleman and Lipton.

To dwell on the latter item for a moment: We avoid PCR, and use only 1 step of polymerase, thus allowing the convenient use of a high reliability polymerase. We do not need to duplicate the whole genome and do not need to rely on the constant use of polystyrene antisense bead and amplify steps, which sound like a big source of possible error and exponentially multiplying losses, and in addition were the biggest percentage of the 7 day lab time requirement for Adleman. We do not require unrealistic annealing of long DNA sequences in which each DNA strand has to pick out its mate from 10^{17} wrong alternatives in the soup. We've demonstrated the capability for error correction. We do not require pre-synthesis of unrealistically long DNAmers. Our recommendation of modified DeBruijn sequences (appendix B) for Turing machine symbols avoids or reduces problems with hairpins and mismatches that would have plagued earlier schemes. Our entire cyclic computational process takes place for DNA bound to a single substrate, and thus presumably immune to loss, in a single receptacle, an important practical convenience advantage.

However, we are still susceptible to unrepairable DNA damage due to restriction enzyme cuts at noncognate sites, contamination (e.g. by stray nucleases; such contamination may be unavoidable), and hydrolysis damage[30]. This situation is exacerbated by the excruciatingly slow speed of the computations we run.

[30] Incidentally, we don't suffer exponentially from at least *some* reaction yields being below 100% because we'll simply be re-running uncompleted reactions during future 7-step Turing cycles, but any parasitic reactionm such as hydrolysis, removing some tiny percentage of our DNA each step will cause exponentially multiplying losses during long computations. Remember that typical computations involve huge numbers of steps

3 Biological significance?

Does the fact that a Turing machine can be made out of DNA have any biological significance? Yes!! Certainly repressor and promotor protein feedback loops (cf. §3.6.1), intron excision, transposon, genetic switch, and antibody synthesis mechanisms (to name a few) ought to give cells considerable computational power. And it is also obvious that in fact cells do use such power to make the little logical decisions crucial to maintaining homeostasis, as well as pivotal growth decisions, constantly. Our particular scheme is very crude in comparison; indeed, antibody synthesis is a computational search task on which cells apparently outperform the best computer hardware and software available today.

But for some reason, this point of view is not very prevalent[31]. We are now going to try to change this. We would like to inspire the new and sometimes useful paradigm of "thinking of biology as a computer programming problem."

First, we are going to show that our Turing machine construction is astoundingly similar to "RNA editing" processes that have been discovered during the last decade, which in turn seem similar to functionalities required by the hypothetical early "RNA life." So this may cast light both on the questions of why RNA editing exists, and also on the question of how life began. Second, we will show that cells face some severe computational problems. Third, we will survey some known biochemical mechanisms inside cells which seem to have computational capabilities, and discuss what those capabilities are.

This is a loosely organized collection of different biological phenomena that look computational, together with some computational models which look biological, and a few new biological hypotheses suggested by our viewpoint. It is hoped that its informational and inspirational qualities will make up for whatever lack of coherence and speculative character it displays.

3.1 RNA editing in Trypanosomes

This is a long story, but we cannot resist recounting it.

The trypanosome *T.Brucei*, which causes African sleeping sickness, has many unusual properties. It is a single cell, $\approx 4\mu$m long, with a flagellum. Its outer wall is covered with glycoproteins which change, apparently randomly, every few days, thus keeping a step ahead of the immune system of the host. Also in the cell are a nucleus, and a single long tubular mitochondrion which contains a so called "kinetoplast" body.

Inside the mitochondrion is DNA, which comes in both "maxicircles" ($\approx$ 30Kbp) and "minicircles" (400-2500bp each; 300-400 kinds, says Correll et al. 1993). The mini-

[31] For example, a quote from the otherwise excellent book (Alberts et al. 1994): "...it is hard to imagine how a cell could keep a long-term account of its division cycles and halt after completing 50." A Turing machine could easily handle this problem, emulate any finite state machine, etc.

circles are interlinked like chain mail. The maxicircles encode at least 13 genes for mitochondrial proteins, but some genes seemed to be mysteriously missing, a large portion of the genome seemed to be noncoding, and some genes seemed to be in the wrong reading frame – very mysterious. Also, the minicircles apparently don't code for any proteins and so their function was initially mysterious too.

These mysteries were resolved by the exciting discovery (Benne et al. 1986) of "RNA editing." According to Maslov 1994, "RNA editing occurs in all trypanosomatid species so far examined," but *T.Brucei* is the most studied one.

Many of the genes in *T.Brucei* maxicircles are in a so called "encypted" form. They are transcribed into messenger RNA (mRNA) as usual. But then this messenger RNA is "edited" over 100 times. For example, the cytochrome C oxidase gene transcript, which is 712 bases long, is edited by the insertion of 398 U's at 158 sites, and the deletion of 19 U's from 9 sites. This editing changes the reading frame with abandon and certainly totally changes the message. At its conclusion, the messenger RNA contains the "plaintext" for cytochrome C oxidase, and is used for protein synthesis. The RNA editing in trypanosomes that has been found so far consists entirely of site specific insertion and deletion of U's, but other organisms have been found which do other kinds of RNA editing[32]. Mutant trypanosomes without RNA editing, die (Stuart 1991).

The editing in *T.Brucei* is performed by "editosome" protein complexes (Göringer et al. 1994, Seiwert & Stuart 1994) with the aid of "guide RNAs." The guide RNAs, which are 35-78 bases long, contain an "anchor" portion 4-14 bases long antisense to the (apparently arbitrary) pattern that they recognize, plus a additional subsequence defining how many (up to 32) U's are supposed to be inserted (or deleted) from (up to 10) sites; they also contain various hairpinned structural portions and a scorpionlike tail containing U's which can be donated to the edited RNA. The gRNA tails can later be recharged with U's by "TUTase" (Simpson 1990) and UTP. The gRNAs will form duplexes with mRNA without requiring protein in some cases.

The guide RNAs are encoded in the minicircles (Benne 1992) and also in intergenic regions of maxicircles.

According to Simpson 1990 "It remains to be seen if the trypanosome type of RNA editing is present in higher eukaryotes." In most genes known so far in most eukaryotes, mammals included, the only RNA editing which goes on (but it happens extensively!) is the excision of introns (substrings) by "spliceosomes." What makes *T.Brucei* and its kinetoplastid kin different is that more than simply splicing out introns is going on – there are insertions and deletions – and also each of these editing operations is apparently coded for by a site-specific and edit-specific "guide RNA," which apparently, in principle, could have specified anything. There appears to be one guide RNA for each edit operation on each cryptogene. Meanwhile in mammals, so far as is known, there are only a few kinds of spliceosomes and although they locate (somewhat) specific sequences, these sequences seem to be few[33].

3.1.1 *We claim RNA editing is a computational mechanism*

The point: this process in *T.Brucei*, specifically, the "guide RNAs," is remarkably similar to the action of the "transition matrix" in our Turing machine construction. Processes like RNA editing are thus capable of Turing universal power and may have evolved specifically to cope with the computational demands of gene regulation in eukaryotes. See §3.6.2 and §3.5.2.

Now in fact, *T.Brucei*'s particular choice of gRNAs is not Turing universal. This is because the U inserts and deletes always lie on the 5' side of the recognition site. This point seems most clearly made in the Maslov-Simpson 1992 paper. *T.Brucei*'s gRNAs seem in general to act on mRNA sequentially and undirectionally, with each editing step often setting up the recognition template for its successor. Also, one is handicapped by the fact that the poly-A tails are on the wrong side (the 3' side). Full Turing behavior would want to allow bidirectional movement and the capability to add an unbounded amount of extra tape.

Probably this choice of disallowing backward branching by *T.Brucei* was "intentional" because full Turing power would have made it difficult to prevent "software bugs" from getting out of hand and would make things much less predictable.

However, it still remains possible that by combining the RNA editing systems of two different organisms (e.g. a *Physarum* and a trypanosome), that we could in fact get a set of string rewrite rules of full Turing power in a single test tube[34]. Cf. §3.6.2.

[32] Examples: The slime mold *Physarum polycephalum* does 54 C-insertion edits on at least one mitochondrial gene. Two unencoded G residues are found in mRNA from paramyxovirus SV5. Maize (Kumar & Levings 1993) edits its C-atp6 gene by performing 19 specific C to U alterations. U→C changes have also been observed in plant mitochondria, while U→A, U→G, and A→G single nucleotide conversions are observed in *Acantha Amoeba Castellanii* mitochondria (Lonergan & Gray 1993). Mammalian apoLipoprotein B comes from an mRNA in which a single C→A, conversion was performed, but this is known (Greeve et al. 1991) to occur via an entirely different sort of mechanism from the one in *T.Brucei*; meanwhile the RNA editing in yeast (*Saccharomyces Cerevisiae*) seems to be more similar to *T.Brucei* (Mueller et al. 1993).

[33] There is also some evidence (antisense RNAs) suggesting that edited mRNAs in *T.Brucei* can be *replicated*. It is also known that RNA site specific endonucleases and RNA ligase are around (B. Sollner-Webb, Nature News and Views 356 (1992) 743.). In other words, the RNA editing processes that we know about may not be the only ones in town.

[34] Also, even unidirectional rewrite rules can have impressive computational power if applied to circular RNA – which exists in nature, see Ford & Ares 1994 and Tsagris et.al 1991.

3.1.2 *We claim RNA editing exists because either it is a computational mechanism, or it is a relic of one*

It has been argued by Landweber and Gilbert 1993 that RNA editing in *T.Brucei* and kin may represent an adaption designed to increase mutation rate and evolvability. Since both mutations in the guide DNA and the cryptogene DNA will cause a mutation, the mutation rate for edited genes, is upped by a factor ≈ 2 (confirmed experimentally by L & G).

We regard this evolvability explanation for RNA editing as illogical. Many biologists have argued (and correctly, as surely L & G would agree) that the reason for various biochemical objects, e.g. DNA repair enzymes, was to *decrease* the mutation rate! If you want to argue that increasing the mutation rate is desirable just in this case, you had better present an evolutionary reason why, for example, *T.Brucei* should particularly want to evolve its cytochrome C oxidase gene more quickly than any other one. L & G gave no such argument. And it is silly to think so. If there is anything an obligate parasite like *T.Brucei* should want to evolve quickly, it is not the internals of its mitochondrion, but rather its external cell membrane – precisely the things that are not coded from mitochondrial DNA. If there is anything *T.Brucei* should want not to mutate, it is cytochrome C oxidase, which (see Stryer) is extremely conserved throughout all known eukaryotes, and in fact is apparently always functional on cytochromes from other species.

So instead, it seems to us that there are two possible explanations for this extensive RNA editing. The first is that it is simply an evolutionary relic. The second is that it serves some important computational function within the cell, the most obvious guess being a control function. Evidence: It is known that life cycle changes in *T.Brucei* are correlated with abundance changes in cryptogene RNAs at different stages of editing: and in the human bloodstream *T.Brucei* turns off mitochrondrial respiration but activates it in insects, although RNA editing is present during both life stages[35] (Stuart 1991).

It is known that gene regulation is a very important and difficult computational problem faced by the cell, cf. §3.5.2. A mechanism with Turing universal (or nearly), and also potentially highly parallelizable, computational power could therefore be a useful thing to have right smack in the middle of the information highway.

As far as the first explanation (evolutionary relic) is concerned, it is known that the kinetoplastids are one of the two evolutionarily oldest lines of eukaryotes (according to DNA divergence evidence, Sogin 1989) and RNA editing in them seems to be "very old indeed, perhaps dating to prebiotic times..." (Benne 1994 p.20). So perhaps, this has something to do with the famous hypothesis of early "RNA life" (§3.2).

[35]This mitochondrial switching another unusual feature of T. Brucei that distinguishes it from most eukaryotes, but it may not be the reason for RNA editing, since, e.g. *T.Evansi* has RNA editing but no functional mitochondrion.

3.1.3 *RNA editing in T.Brucei is a Turing machine and this fact may be exploitable to make a fast in vitro computer*

To summarize the last two subsubsections: RNA editing is a natural way to construct a Turing universal computer, and its location in the biochemical information pathway is a natural place for *T.Brucei* to want one.

Now, Seiwert & Stewart 1994 have managed to prepare a *T.Brucei* mitochondrial extract, containing intact editosomes, and used it to do one particular edit step of an synthetic mRNA in vitro. In the event that one could clone editosomes, and also in the event that gRNAs really are a general purpose specification for arbitrary edit operations as is currently thought, we could, by placing editosomes, an energy source (ATP and UTP, probably) gRNAs, a poly-adenylatase and TUTase, and a mRNA "tape" in a test tube, get a second way to make a Turing machine in vitro.

This seems likely to work, although since it is obviously undesirable to work with the organisms that cause African sleeping sickness, it may be best to switch to another trypanosome. This Turing machine, which would really be better describable as a rewrite rule system (cf. §3.6.2), would have two advantages over our construction. First, it would probably be a lot faster: although we are unaware of any measurement of editing speeds in *T.Brucei*, if one assumes that *T.Brucei* mRNAs have about the same halflife in vivo as mRNAs in other eukaryotes (a few minutes), then editing at 158 sites is performed per mRNA in a few minutes, for a rate of about 1 edit operation per mRNA per second. That is 10^3-10^4 times faster than our (& previous) constructions with present day standard lab techniques, and would allow $\approx 10^5$ rewrite operations per day per mRNA. Second, since the underlying computational model would be a string rewrite system (see §3.6.2) rather than a Turing machine, additional parallelism would be obtainable. But unfortunately, since RNA is less stable than DNA to backbone hydrolysis (Lindahl 1994) especially in the presence of ions such as Mg^{++}, this computer would dissolve even more quickly than our DNA based construction.

3.2 *"RNA life" ideas of the origin of life*

The orthodox picture of present day life has DNA, which carries the information and replicates, RNA, which transmits the information from the DNA to the ribosomes, ribosomes, which synthesize proteins according to the commands on messenger RNA, and proteins, which do the work of the cell – whether it be chemical (enzymes), mechanical (actin and myosin; microtubules; cilia) or structural. There are also lipids, which form the cell membrane, but their synthesis and destruction is overseen by proteins. Finally there is ATP (and sugars), which are the currency of energy in the cell. All these components are highly specialized and optimized for their tasks.

In the early days of life, though, things could not have been this sophisticated. One well known speculation (Miller & Orgel 1973, 1974; Joyce 1989) is that in an earlier stage of life there was just RNA. The RNA played both the genetic role currently played by DNA, and the enzymatic roles currently played by proteins, and as for an energetic role, of course ATP is highly related to RNA. Since RNA viruses exist, it is known that RNA can serve to carry genetic information. Recently, pure-RNA enzymes (Kruger 1982, Guerrier-Takada 1983) have been discovered (the tRNAs, ribosomal RNAs, and telomere template RNAs had been known for a long time to be co-enzymatic, Cech 1994) so it is known that RNA, unassisted, can play enzymatic roles. (What about lipids and the cell membrane? Not clear. Perhaps originally no membrane was needed since life was just a single replicating RNA molecule?) RNA can form complicated structures stabilized by many self-hairpin pairings. Nucleotides and their phosphates (e.g. ATP) are known to arise in "primordial soups" (Joyce 1989). Recently, evolutionary selection has been repeatedly demonstrated by an in vitro, selective and error-prone, RNA replication process, leading, after 10 generations starting from a random sequence pool, to a multimillion-fold enhancement in the particular enyzymatic activity being selected for (Ellington & Szostak 1990, Lorch & Sostak 1994, Sassanfar & Szostak 1993) including the de novo synthesis of an all-RNA RNA ligase (Bartel & Szostak 1993).

Pace 1991 has pointed out that all-RNA life would have been very unstable under the severe hydrolysing conditions that probably existed over most of the primordial earth, and therefore perhaps "RNA world" was preceded by some earlier and even simpler form of life; a stabler artificial polymer called "PNA" (peptide nucleic acid, Nielsen et al. 1991) may fit into that picture.

Anyway, it has been shown that RNA molecules, acting alone, are capable of catalysing RNA cleavage and joining[36] (and self splicing, self cleaving, and self circularizing; see Cech 1987, Kruger 1982, Uhlenbeck 1987). An entirely enzyme free system has been shown to be capable of "high" fidelity RNA polymerization guided by RNA templates 6-10bp long (Inoue and Orgel 1983; admittedly no such system has been demonstrated with racemic nucleotide mixtures).

3.2.1 The idea that one can make an RNA Turing machine should be important for "RNA life" ideas of the origin of life

The connection between all this and RNA editing and our Turing machine is just this: we will demonstrate in §3.6.2 that it is easy for systems capable of RNA editing like operations, to get Turing universal power. With all this evidence, it seems quite possible for an all-RNA system to evolve with Turing universal power. Thus, computer science may in this case be providing insight into how life could have arisen.

[36]Incidentally, DNA in single stranded form does not exhibit catalytic activity – Cech: "the importance of being ribose."

3.3 Introns, Transposons, and junk

Transposons are subsegments of your DNA which can transplant themselves from one location within your genome to another. *Replicant transposons* (which we will call "replicons") can do more than just move – they can in fact make copies of themselves which implant themselves elsewhere in your genome. *Introns* are substrings of your genes, 10^1-10^4bp long, which are transcribed into mRNA but then excised by *spliceosomes* before that mRNA is used to direct the manufacture of proteins. (The non-excised portions are *exons*.) 97% of the human genome is *junk DNA* which has no known function – and since much of it consists of short repeating sequences, it cannot be encoding very much information.

All of these things seem to be at best a large waste of resources and at worst fatal. Fatal because it is known that transposon hits can deactivate vital genes and thereby kill the cell – and more than half of the observed mutations in *Drosophila* are due to transposon hits.

Indeed, replicons seem to be taking over the genomes of higher life forms. 10% of the human genome is currently made of copies of the two most popular kinds of replicons, called "Alu" and "L1." There are 10^6 copies of Alu, and each one apparently serves no function other than directing the synthesis of proteins important for its own copying!

Meanwhile in bacteria such as *E.Coli* there is almost no junk, few if any introns, and the only known replicons are phages (such as μ-phage) which tend to be fatal and are not present in healthy bacteria. Even *E.Coli* has $\approx$ 10 transposons, however. Some of these transport themselves to apparently random sites, some to hotspots, and some only to sites with special recognition sequences such as `NGCTNAGCN`. The transposons in *E.Coli* seem to move at rates of 10^{-4}-10^{-3} migrations per transposon per cell per generation.

Transposons can be "autonomous," which means that they encode all the proteins necessary to carry out their migration and/or replication or "nonautonomous," which means they have to parasitize proteins from other parts of the geneome, possibly including other transposons. All replicons known so far use a reverse transcriptase, similarly to μ-phage. Replicons may be viruses which have lost their coat protein genes and consequently the ability to exist outside the cell (although the other hypothesis is that viruses evolved from replicons!) and nonreplicons may be replicons which have lost the ability to multiply exponentially, but still persist through evolutionary time because they confer some advantage to the host.

3.3.1 Why do replicons, junk, etc. exist? The old explanations and their problems

What evolutionary advantages, if any, do transposons and junk confer? Why are they there? Transposons can have uses. Genes located between two transposons (or within a transposon) can get moved around in the genome, and also copied, and similarly exons can get shuffled by transposon action. On evolutionary time scales, the combination of transposon and introns can provide a mechanism for generating useful mutations. A species without transposons and only evolving by point mutations might take virtually forever, say, to evolve three different rhodopsin genes in order to acquire 3-color vision. It would be out-evolved by a creature whose replicons happened to make three copies of its rhodopsin gene, and then perhaps some of the exons got shuffled by nonreplicant transposon action... and the three rhodopsins then, finally, got tuned by point mutations to be sensitive to different colors. The point is that reshuffling and copying gene building blocks which are known to be useful, is much more likely to evolve a useful new gene than random point mutations alone. Indeed, creatures that reproduce sexually go through a stage called "crossing over" in which, conceptually, substrings of the maternal genome are swapped with corresponding substrings of the paternal genome, with the boundaries of the swapped regions apparently being chosen at random. The crossover process undoubtably arose because it causes variability in the genome which has a comparatively large chance of being useful and a comparatively small chance of being harmful. Perhaps the evolvability advantages of creatures whose genomes were equipped with transposons, outweigh the dangers.

On much faster time scales, of a single lifetime, transposons, or mechanisms similar to them, also can have uses. Transposons are suspected to be part of the developmental and regulatory process in the maize plant. (Obviously, by moving promotor and repressor subgenes around, regulatory and developmental effects are achievable. Indeed, the nematode *Ascaris* reshuffles its whole genome during development.) As discussed in §3.5.1, reshuffling of DNA segments is an essential part of antibody synthesis in vertebrates. The amplification of genes made possible by mechanisms like replicons causes certain genes in *Drosophila* salivary glands (presumably genes related to making saliva) to be present in over 1000 copies. These genes are present only in much smaller numbers in normal *Drosophila* cells. Similarly, *Drosophila* ovary cells have genes which are greatly amplified.

Do not be convinced that transposons and introns arose because of all the advantages they confer. Amplified genes are also known to be present in mammalian cancer cells, which suggests that this mechanism is dangerous – cancer is a leading cause of death in humans. Human leukocytes have a higher risk of cancer than other human cells, which is probably related to the fact that they shuffle their genomes (cf. §3.5.1). *Drosophila* males are known to have lifespans that are shorter, the more P replicons they have (Woodruff 1992, see also Driver & McKechnie 1992), presumably due to an increased load of somatic mutations. Meanwhile it is known that silk moths, some of whose cells produce huge quantities of silk protein, do *not* amplify the silk gene, which indeed is present in only one copy. In fact most eukaryotic genes are present in only one copy. That suggests that gene amplification, while perhaps useful, is not necessary. There is also some evidence that exons are not especially useful gene building blocks and that the intron borders occur rather randomly, but all such evidence so far is hard to interpret and cannot be said to be conclusive.

Finally, the biggest problem of all: replicons are very bad news. It has been observed that the most successful replicons (e.g. L1) have negative feedback mechanisms that prevent them from multiplying too rapidly. (Still, their growth is assuredly exponential over evolutionary time scales.) This is because any transposon which multiplied too rapidly would take over the genome and kill off the species. But, suppose that the replicon multiplies slowly enough that it can contaminate the genome of every member of that species, before it grows large enough to have any injurious effect. (Alu has managed that.) In that case, that species seems automatically to be an evolutionary dead end. And it furthermore seems to us that such replicons will always be there, since evolutionary processes favor the replicons with the highest growth factors that are possible without death. So, replicons ought to make higher life forms impossible.

3.3.2 A "computer science explanation" for replicons and junk

Here is our interpretation of all this. Higher life forms had to have an immune system (cf. §3.5.1), so they had to have gene reshuffling or some other highly computation-like ability in order to outcompute their simpler parasites. So Turing universality was inevitable – not only for this reason, but also because of the huge evolvability and developmental/regulatory advantages that it allowed. But once the necessary mechanisms for Turing universality became widespread, life became very dangerous. As is well known, it is very easy to change a computer program to make it do something unpredictable, nonlocal, complicated, and possibly destructive. In particular, replicant computer programs are possible and, as is also well known, can be very dangerous – and that is exactly what replicons are. The same sort of gene shuffling mechanisms used by vertebrates to fight infections are known to be used by eukaryotic parasites to synthesize coat glycoproteins which keep changing. There is thus a computational arms race between hosts and parasites and with the advent of Turing universality it in fact becomes undecidable whether, e.g. something inside your genome is harmful... So for this reason, there was incentive for cells to try *not* be be Turing universal, or at any rate to try to keep their computational capabilities under as tight control as possible.

We have argued that replicons, if left to themselves, ought to make higher life forms impossible. But higher life exists. It seems to us that the only possible solution to this paradox is to infer – or predict – that replicon "police" exist. Suppose some replicon were to start be-coming a heavy burden, occupying, say, 99.9% of the genome. Obviously, if some "killer" gene were to evolve which produced enzymes which recognized that replicon, excised it from the genome, and destroyed it, then members of the species which had the killer gene would have an advantage and the replicon would be wiped out. On the other hand, once the replicon was gone or greatly reduced, this killer gene would be a very dangerous thing to have, since it produces DNA destroying enzymes. It would quickly vanish[37]. Because the most prolific transposon would tend, as it grew exponentially, to destroy its opposing less prolific transposons (but not destroy essential genes), over evolutionary time, by the time it became necessary to call in the police, only one replicon would desperately need to be killed, and the cure therefore would work.

Unfortunately, the replicons can fight back. One way is to camouflage themselves as essential genes. The Alu replicon strongly resembles the essential 7SL gene, which is usually interpreted as evidence that it evolved from that gene, but could also be interpreted as the self defense measure that explains why Alu is still around.

We further speculate that junk DNA may be in large part composed of the corpses of former replicons, which would explain (1) how it got there, (2) the fact that a lot of junk is highly repetitive (Britten and Kohne 1968)[38] and (3) our theory is also supported by the fact that closely related species often have vastly differing amounts of DNA (factors of 5-10).

Finally, even with replicon killers, higher life forms still face the problem that their genomes may gradually become overwhelmed by junk. Oddly, it may be that nonreplicons may provide a way to keep down the junk, since such a transposon could, by failing to reinsert itself, serve to delete a piece of DNA. If the deleted segment were junk, this deletion would be harmless, indeed slightly beneficial.

[37]That is perhaps why no replicon-killer gene has been seen so far and this is all speculation. But it is still provable. If it were shown that the Alu replicon were not present in monkeys, that would prove it appeared after the divergence of humans from monkeys – and grew to 5% of the genome in that comparatively short time. The only way such growth rates could be possible and higher life forms could still exist, would seem to be killers. Actually Alu variants exist in many mammals, although not in frogs and flies even though they have similar 7SL genes, and appear to have arisen about 65 Myrs ago (Deininger & Daniels 1986). This still seems sufficient to make our case, since (1) the Precambrian metazooan "bio-explosion" occurred ≈ 530 Myrs ago and (2) Alu's proliferation from 1 to 10^6 copies per genome in 65 Myr implies a doubling time of 2 Myr, which seems to leave the human genome with at most 9 more Myr to go before Alu takes over 100%. It is also known that the P replicon in *Drosophila* has appeared, and multiplied by a factor 2-50, in only 50 years.

[38]It has also been speculated that such short repetitive sequences as 5′-ATAAACT-3′, which form 25% of *Drosophila* DNA, may serve some structural function, but this function is probably not very important since even closely related species often have totally different repeats.

3.4 The secret lives of hypotrichous ciliates

Hypotrichous ciliates (single celled eukaryotes such as *Euplotes Crassus*, see Prescott 1992 and Tausta et al. 1991) have two micronuclei (diam$\approx$ 4μm) and two macronuclei (diam$\approx$ 25μm). Each micronucleus contains what at first appears to be a fairly typical looking eukaryotic genome.

However, following mating (that is, exchange of a pair of haploid micronuclei which fuse to make normal diploid micronuclei; this is comparatively rare, usually reproduction is asexual), all nuclei are destroyed except for the new micronucleus, which is copied. The new micronucleus then gives rise to a new macronucleus. During the construction of the new macronucleus, remarkable events happen. The DNA from the micronucleus is chopped into a large ($\gtrsim 5 \times 10^4$) number of short (smaller than a gene) segments; some of these segments are removed, some are destroyed, and some are ligated – often in a different order. For example, the 1.5Kbp actin I gene is cut out of the genome and chopped into 17 pieces. Three of these pieces ("*bcd*") join to form a 175bp circle, one ("*a*") forms its own 23bp circle, and the remaining 13 pieces are permuted according to

$$...3a4b6c5d7e9f2g1h8... \to sf987654321ghe,$$

where s is a 15bp spacer which is apparently synthesized de novo, to get a correct (transcribable) actin I. (The original pre-permuted actin I gene wouldn't have worked for directing the synthesis of actin.)

After such operations are complete, and huge amounts of non-gene DNA (i.e. $\approx 95\%$ of the genome) are destroyed, one is left with a large number of DNA segments (200-15000bp), each of which contains a single transcription unit.

These are then equipped with standardized telomeres which are also synthesized de novo, and then amplified by ≈ 1000 fold replication. Finally, the finished macronucleus divides into two macronuclei, each of which contains about 2×10^4 gene-sized species of DNA, each species coming in ≈ 1000 copies, and the cell continues its merry way, with two macro and two micronuclei. The macronuclei are used for all transcriptional purposes and the micronuclei for apparently no purpose, except for storing the genetic information for eventual use during mating and the production of new macronuclei.

How and why[39] is all this done? All we can say is, this is quite amazing.

[39]Our conjectural explanation for "why" is that this is a "preprocessing" step. In computing it is well known that time spent beforehand doing a precomputation (e.g. sorting a phone book) will often more than pay for itself later. Similarly, by preprocessing its DNA to remove introns and junk, *E.Crassus* is able to avoid the later metabolic cost of manufacturing all the intron mRNA which is wasted by other eukaryotes. By retaining introns in its micronuclear genome it still gets whatever evolvability advantages they confer.

3.5 The cell as a computer

Cells are known to face and solve some difficult computational problems.

3.5.1 Antibody synthesis

There are $\approx 10^{12}$ immune system cells in your body, which is at least an order of magnitude more than the number of your neurons. Arguably your immune system cells are doing a lot more computation than your brain.

What does your immune system do? It detects foreign substances, ("antigens") and then designs, de novo, an antibody (protein) molecule that binds to that antigen. Antigens include "any inorganic or organic molecules you care to mention... when attached to a [suitable] carrier molecule," according to Jerne 1984, and that includes molecules just synthesized by chemists which have never existed before. Once the design process is completed, the antibody is synthesized in comparatively large quantities, circulates throughout your body, and binds to antigens. Killer "T" cells then rove throughout your body, find things which have antibodies attached to them, and kill them, e.g. with digestive enzymes. Similar systems are in all vertebrates.

Many details of the design process are now understood[40], and here is the rough picture.

Antibodies consist of a standardized Y shaped portion and variable portions near the "hands" of the "Y." The variable portion is coded for by a DNA region containing some of the 200-300 antibody building block subsequences. "Resting B-cells" use recombinases to continually shuffle their antibody building block subsequences into and out of the coding region. The recombinases recognize a specific heptamer and nonamer upstream and downstream of each building block. With 200 blocks, of course $\approx 200!$ permutations would be possible, but even more variation is possible since the building blocks can also get shifted within prescribed ranges and immune cells also exhibit point mutations (in this DNA region) at a much higher than usual rate ("hypermutation"). Thus a region of the DNA of B-cells is continually changing. If and when a moment occurs when this DNA happens to code for antibody which binds to the antigen, the B-cell is stimulated to cease exploring DNA-space and instead to reproduce. Its descendants serve as factories each outputting 2000 antibody molecules per second. But some fraction of these descendants continue to explore DNA-space in order to fine-tune the antibody's antigen affinity; thus the antibody design and synthesis process is an example of *Lamarckian* evolution going on inside your body in an effort to solve a combinatorial problem by an exponentially large heuristic search.

Most simple parasitic organisms cannot compete with the huge parallel computational capability implied by our 10^{12} immune cells, which is why we can recover from many diseases.

Lamarckian evolution will also rear its head in §3.5.4.

3.5.2 Gene regulation

The prokaryotes (cells without nuclei) seem to have an upper limit of about 6000 genes, and this may in part be due to the problems inherent in controlling that many genes with the limited mechanisms (repressor and promotor proteins) available to them. Eukaryotes have many more genes than this. Humans are estimated to have 10^5 genes, and there are at least 5 regulatory elements per gene. (Robert Tijan, quoted in Science 263 (Feb 1994) p608.)

Certainly the usual promotor and repressor (proteins binding to DNA near the gene they regulate) mechanisms are used extensively by eukaryotes as well as prokaryotes. The *Drosophila* Eve gene has a 2×10^4bp prologue, used for regulatory purposes, which binds > 20 regulatory proteins, and this is small potatoes compared to some mammalian genes. But what do humans and other eukaryotes have that prokaryotes do not, that allows their cells to handle the huge excess regulatory burden?

One answer may well be: introns. Prokaryotes cannot use intron splicing since they have no nucleus to physically separate incompletely edited RNA from ribosomes. It is known that an intron in the roundworm *C.elegans* codes for an RNA that binds to and deactivates the mRNA for another gene. It is also known that certain aminoglycoside antibiotics deactivate certain RNA intron splicing processes (von Ahsen et al. 1992), which suggests that these are important processes. The fact that intron excision is done by a multiprotein spliceosome complex also suggests its importance. Nevertheless, at the present time, most introns have no known function and ignorance is widespread.

In principle, RNAs could bind to mRNA introns, and excised introns could bind to other mRNAs, and some of these bindings could either deactivate or activate spliceosome or RNAase actions. This could set up a complex network of possible interactions which could serve a computational and regulatory function operating on fairly short time scales. (The half life of mRNA in rat liver cells is known to be a few minutes.) This is the usual sort of speculation about the possible function of mRNA splicing in eukaryotes. What is the computational power[41] of such a network?

[40] The related problem of how the immune system learns what substances are "friend" and which are "foe" is not understood. Antibodies themselves can serve as antigens, leading to the production of anti-antibodies. At any given time there are $\approx 10^7$ different kinds of antibodies, totalling $\approx$ 100grams, circulating in your body, as compared to the $\approx 10^5$ proteins coded for in the genome we were born with, and (Jerne 1984): "...in its dynamic state our immune system is mainly self-centered, generating anti-idiotypic antibodies to its own antibodies, which constitute the overwhelming majority of antigens present in the body. The system somehow maintains a precarious equilibrium with the other normal selfconstituents of our body, while reacting vigorously to invasions..."

[41] AND, OR, and NOT functionalities would seem thinkable which would imply "Boolean completeness."

3.5.3 Assembly of complex structures; development of multicellular organisms

How can it be that all of our cells have the same DNA sequence[42] but we have many different kinds of cells? How do our muscle cells, retinal cells, nerve cells, bone cells, fat cells, kidney cells, and so forth[43] know which to be? How do they all manage to assemble themselves into a coordinated structure? Neurons are single cells whose axons can be meters long, and these axons somehow thread themselves between all your other cells to go to the right places. All this is the problem of multicellular development and differentiation, one of the "great unsolved problems" in biology.

A perhaps related question is: how do complicated structures *inside* cells get assembled? It is known that if the component parts of some virus particles or ribosomes are placed in buffer, these complex assemblies will magically self-assemble. Many proteins magically self-fold into the correct shape. But more complicated cellular structures will not self assemble from their components. For example, entire cells will not re-assemble, after they have been destroyed... and so far as is currently known, nor will complex subcellular structures such as cilia, mitochondria, and centrioles.

As far as the assembly of complex subcellular structures is concerned, simply providing the components in the right temporal order, rather than all at once, may go a long way. This is something that finite state machines, or the more powerful models of computation we describe in §3.6.1, are easily able to do. Similarly to the way that construction crews assemble large buildings, temporary "scaffolding" may be required which is later destroyed. Thus it is known that some bacteriophage heads are assembled with the aid of a second protein which does not appear in the final phage heads, which therefore won't reassemble if they are disassembled (Mathews et al. 1983). It may also be that the transportation system in the cell has to be more sophisticated than is currently supposed. Similarly to the way that ribosomes are general purpose protein constructors which work to order to mRNA blueprints, the microtubule vesicle-transportation system of eukaryotic cells may also work to order, according to some sort of "train tickets." It is a commonplace that membranes in cells have many sophisticated kinds of portals, and it is known (Lombardi, Soldati, Riederer 1993) that vesicles "dock" with the right lipid membranes with the aid of recognition molecules called "Rab proteins."

Be that as it may, the entire problem of multicellular development is easy to explain, assuming we live in a certain oversimplified mathematical model universe – a "cellular automaton." See §3.6.3.

[42] Or at least, that is the orthodox picture; of course, our immune system and the phenomenon of gene amplification are counterexamples to this picture.

[43] Alberts et al. 1994 give a catalogue of 210 major categories of human cells.

3.5.4 Protection and Lamarckian evolution

The orthodox ("Darwinian") view of evolution is that mutations are entirely random, and the ones which happen to confer increased fitness, are the ones which stick around. "Lamarckian" evolution, is the hypothesis that organisms somehow sense what changes need to be made, make them – more or less "intentionally" – and then pass them on to their offspring. It is clear that Lamarckian evolution, were it feasible, would confer a great Darwinian selective advantage... so that what one should expect is some compromise between Darwin and Lamarck. While it may not be possible to choose intentionally which mutation you want, in many circumstances it may at least be possible to bias the Darwinian dice to make mutations which are more likely to be favorable, more frequent, and mutations which are more likely to be deadly, less frequent.

Meanwhile in computer engineering, it has long been agreed that, in order to build robust computing systems, varying degrees of "protection" and "damage limitation" are essential. The "operating system" and its work areas are generally highly protected – by hardware – against modification by user code. Indeed, some machines have varying levels of protection within the operating system. Also, certain useful things the computer can do, e.g. disk I/O, user programs are *not allowed* to do – unless they work through calls to operating system subroutines which decide whether they will deign to permit it. All this protection in fact *reduces* the accessible computational power of the machine and the creative freedom of the programmer, but it is nevertheless regarded as essential for robustness in the real world of bug-filled programs.

By analogy, we argue that there ought to be varying degrees of protection against various kinds of mutations in different parts of your genome. The most essential and evolutionarily oldest parts ought to be the most protected, and the parts which seem most likely to have something to do with your current problems (e.g. highly expressed?) should perhaps be more mutable. In addition to this sort of varying spatial protection, one would want varying temporal protection – the best strategy is to choose to mutate more when you are under selection pressure. Of course, when one is under such pressure, one may be *forced* to mutate more – a starving lifeform may not have the resources to correct genome damage and conduct high accuracy replication – but it would be better to try to mutate intentionally before this point is reached, because that way you will be more able to control both the rate and the type of mutations you will incur.

Suffice it to say, there is evidence in support of all of this. We have already argued (cf. §3.3) that transposons and sexual evolution (and also genome "crossover") may have arisen in order to make more favorable mutations more likely. The fact that organisms under selection pressure mutate more, and in particular have more transposon mutations ("transposition bursts"), which are in some sense "intentional," is well known. (And 80% of *Drosophila* mutations are due to transposon hits.) Genes, such as the famous p53 gene, exist which can exert control over the transposon rate – when p53 is turned off, which often happens in cancerous cells, this rate can increase by a factor of 10^5. It is known (Kricker, Drake, Radman 1992) that methylated dimers can serve to protect DNA regions, in particular important mammalian genes, against transposon hits.

There is some evidence, though not yet as convincing as the above, that more important genes mutate less often during replications. Especially, there is evidence that *E.Coli* with defective lactose processing genes, will mutate more often, in different ways, and more than usual in the Lac gene region when exposed to lactose selection pressure. When stressed in the absence of lactose, this bias vanishes. Some recent experiments by Cairns and by Foster have been very convincing in this line (Foster & Trimarchi 1994; Cairns, Overbaugh, Miller 1988; Cairns & Foster 1991, 1992).

3.6 Models of computation in vivo

3.6.1 Orthodox bacterial model

A computational model we dreamed up to encapsulate and simplify the internal operations of bacteria (or at any rate, the orthodox picture thereof), say, is as follows.

Bacterial model of computation.

1. You have an infinitely long "tape" which is read-only. Each tape square contains one of 4 characters. In addition, characters may be of two colors, red or blue. The tape contains the "program" and "data" and its initial state could be encoding an infinite amount of information, but let us say that initially only a finite number of readable substrings of the tape are red.

2. A computational step is: you find a all-red substring S of the tape which is long enough to contain a special "start" and "end" subsequence. This substring specifies another substring S_2 and one of two actions to take:

A Find all substrings in the tape which are the same as S_2 and which are currently all-red, and color them all-blue.

B or: Find all substrings in the tape which are the same as S_2 and which are currently all-blue, and color them all-red. (Substrings which are partly red and partly blue are unaffected.)

3. The rules are: if a computational step is possible, pick one of the possibilities at random and do it. Keep going forever (or until no step is possible; noninterfering computational steps could be run in parallel).

Obviously, the "tape" represents the bacterium's DNA, the 4 symbol alphabet corresponds to the 4 nucleotides, and the two colors blue and red correspond to "bound by a repressor protein" and "not." The fact that the tape is read-only is the orthodox picture of bacteria.

The question is, what is the computational power of this strange model. If the tape is finite, the situation is boring from the computer science point of view: the situation is just a finite state machine.

If the tape is infinite, Hava Siegelman showed us rather simply that the bacterial model of computation would have super-Turing power. The proof is simple. Prewrite, in order, an uncomputable sequence on the tape, in the format $1, f(1); 2, f(2); 3, f(3); ...; n, f(n); ...$ where $f()$ is some uncomputable function. The ;'s contain start and end sequences and the integers n are colored blue initially. Then, to read out $f(n)$ in two steps, paint n red in step 1 and read $f(n)$ in step 2.

Of course this super-Turing power is not really accessible biologically, but this little example does start the mental wheels turning.

With a rather more simply preformatted tape, namely just a periodic sequence, we still claim that fully Turing power is possible, *provided* that the model is extended so that binding proteins will only bind to (or be removed from) their DNA recognition sites *if* said site is sufficiently close to a previously bound protein which is a promotor of such activity. ("Eucaryotic gene regulatory proteins often assemble into small complexes on DNA" – Alberts et al. 1994.) We claim that a finite number of such proteins and relationships among them will suffice to emulate a universal Turing machine and with only constant factor slowdown.

The proof is omitted, but straightforward. For example one can construct an emulation of a Post "tag system" as in Minsky §14.6.

3.6.2 A computational model based on RNA editing

Suppose we have mRNA, which we model again as a tape with 4 symbols, and we have a finite set of rewrite rules which specify certain context specific single nucleotide conversions, insertions, or deletions, or which can splice out an entire intron (substring) provided it has specific recognized start and end sequences, in one step.

Any such system of rewrite rules may have ambiguities, i.e. several rules may be applicable at any time, in which case we break ties temporally at random.

Again we claim this model of computation has full universal Turing power. (Of course, this model is extremely similar to Chomsky grammars and Post rewrite systems.)

Although all of these editing capabilities are known to exist biologically, they are not presently known to exist all at the same time in the same organism. To make matters realistic, then, we proclaim that in fact,

> Only a finite collection of rewrite rules which insert or delete single U's in specified contexts of an mRNA sequence will suffice to get Turing universality;

no nucleotide conversion rewrite rules, or insertion or deletion of non-U's, are needed. This is compatible with current knowledge of *T.Brucei*. We do need an infinite mRNA "tape" however, prewritten with a certain simple periodic pattern (poly-A will suffice) except in some finite portion, or equivalently we need the capability to append an unbounded number of A's, say, to one end of our mRNA. (Yes, only A's and U's are needed.) Eukaryotes are known (e.g. Baker 1993) to have mechanisms for appending long poly-A tails to the 3' end of mRNA.

The proof of this can be accomplished by embedding a 1-dimensional Turing universal cellular automaton with a two character alphabet and a neighborhood notion 18 bits wide in the *T.Brucei* model, where the tape is a periodic sequence in which, in each period, a U can either be present or absent in specified locations. Alternatively one can construct a universal Turing machine directly with certain special head states and tape square states.

3.6.3 A cellular automaton modeling multicellular development

A "cellular automaton" (CA) is a model of computation in which the world is an infinite square (in 2D, or cubical, in 3D, or at any rate periodic) grid, and each square contains a single character from a finite alphabet. All cells are updated synchronously to become, at time $t+1$, a new state which is a function (the "transition rule") of its previous state at time t, and that of its immediate neighbors.

The "game of life" is a 2D cellular automaton invented by J.H. Conway, described by these incredibly simple rules: cells have two states, "alive" or "dead," and live cells stay alive at time $t+1$ if and only if they have 2 or 3 live neighbors at time t, and a dead cell turns alive if it has exactly 3 live neighbors. In Conway's book (Berlekamp et al. 1982) the game of life is shown to be capable of Turing universal power[44].

Biologically, we may state this as follows: **Cellular automaton model:** at any time, depending on its internal state, that of its immediate neighbors, and an internal random number generator, a cell may choose, according to a finite set of allowed transition rules, either to divide in a specified direction, or to change state, or to die (vanish).

We claim that this model, entirely unaided, is easily capable of duplicating the complicated sorts of differentiated behaviors found in real multicellular lifeforms. Unlike Conway's game of life, though, the cellular automata we are concerned with here have potentially very large state and transition rule sets, since the only constraint – that their specification be small enough to fit into a eukaryotic genome – is rather weak.

We claim that, if the transition rule is sufficiently complex (≈ 40 bits of state information per cell should suffice – a number of bits of course dwarfed by the size of huge eukaryote DNA's), then it is easy to write a program (i.e. state transition rules) so that a single cell, initially alone in the universe, will develop into essentially any complex structure whatever.

Also easily, this program can be made to include self correction mechanisms to make it fairly robust to unintended random cell deaths and mutations.

Testable predictions would include: the existence of messenger chemicals and channels which would allow cells to know about the state of their immediate neighbors (also one could consider extending the model to allow cells to broadcast chemical messengers which would diffuse throughout large regions, but this is not necessary); the existence of crude "clocks" so that cells could keep track of time; and the prediction that a cell in the same state and with the same neighbors will always do the same thing.

Also, the entire transition matrix need not be specified, instructions only need to be specified for those neighborhood structures which are likely to arise in practice. This is perhaps the explanation for "teratomas," which arise from grafting developing fetal tissue into unnatural tissue locales, the result being a "bizarre growth... a disorganized mass of cells containing many varieties of differentiated tissue – skin, bone, glandular epithelium, and so on – mixed with undifferentiated stem cells that continue to divide..." (Alberts et al. 1994).

Are the true mechanisms of multicellular development actually of this sort? Evidence: It is known that some simple multicellular organisms do develop according to rules which entirely fit inside the cellular automaton framework. For example, Mitchison and Wilcox 1972 investigated *Anabaena Catenula*, a simple organism consisting of 1-dimensional filaments of cells of 4 visually distinguishable types, which we shall call $\{A, B, C, D\}$. The transition rules $A \to BC$, $C \to BC$, $B \to DA$, $D \to DA$ seem to hold whenever one cell divides into two. "We have tested this rule in more than 600 individual cell divisions and have found that it holds without exception," they wrote.

The entire life cycle of the hermaphoditic transparent roundworm *Caenorhabditis Elegans* has been studied to the point where the entire ancestral tree of every cell's life (each of the 959 somatic cells has a name) is known: every cell divides into cells of a specified type according to a unique schedule. But the situation must be more complicated than this in higher organisms – certainly each of the $\approx 10^{13}$ human cells cannot have its own name encoded by the original zygotic DNA, since that DNA is only 3×10^9bp.

It is known that tissue cells in both insects and mammals have some idea of their position within the body, and cells whose state corresponds to one such location can still behave consistently with their former state if they are artificially transpanted to a different location. Alberts et al. 1994: "One of the most remarkable revelations of modern genetics has been that almost all animals seem to use the same highly conserved... machinery to record positional values along the head-to-tail axis of the body... discontinuities of positional value provoke local cell proliferation, and the newly formed cells take on intermediate positional values so as to restore continuity...

[44] Admittedly their game of life Turing machine emulation has an exponential slowdown ratio, but almost certainly the slowdown need only be polynomial and this was only an artifact of Conway and co.'s desire to keep the proof simple.

the molecular mechanisms that underlie this crucial form of growth control are unknown."

A *morphogen* is a substance whose concentration is read by cells to discover their positions relative to "landmark," or "beacon" cells. Alberts et al. 1994: "Morphogens are thought to be a common way of providing cells with positional information or controlling their pattern of differentiation, although there are still only a few cases where a morphogen has been identified chemically." These are to be distinguished from "hormones," which are transmitted bodywide and whose concentration gradient, if any, is not thought to be of interest.

We disparage the morphogen hypothesis, despite the fact it was proposed by our hero Alan Turing. Certainly morphogens are important for symmetry breaking in early fetal development, and perhaps elsewhere (for example, perhaps the growth of capillaries toward oxygen-poor tissue is stimulated by a morphogen secreted by such tissue, or perhaps oxygen itself is the morphogen). But, morphogens are inherently inaccurate and error prone, because they are analog rather than digital. They are also incompatible with the sharp digital boundaries that appear everywhere in organisms (e.g. the surface of a bone). Threshhold mechanisms have been proposed by morphogen fans to get around that problem, but it isn't that simple... such problems do not arise in the cellular automaton picture.

The positional-address genes, or at least, the few that are currently known, are digital.

Consider neuronal axon growth – axons grow along a precisely defined path, sending out and retracting filopodia to explore the regions ahead before deciding on their growth path. (Later neurons often simply follow parallel to a trailblazing axon, though, resulting in a neuronal bundle.) Also it is worthy of note that before sending out their axons, neurons first migrate to a precisely defined location.

In conclusion, one may conjecture that the mechanisms of multicellular development resemble, computationally, the "cellular automaton" picture and that the "morphogen gradient" picture is inadequate. The above facts may not prove it, but they do make this plausible.

It also seems very plausible that the algorithms encoded by cells's DNA must have some sort of "higher level language" allowing entire subroutine calls such as "build an entire arm" to be encoded in a small space. The very existence of people who have functioning extra fingers, say, seems to prove that. So, it seems to us, looking at the problem of multicellular development from the computer science point of view, that experimentalists's attention should be devoted to figuring out the the transition matrix, the chemical messengers, the clock[45], and the higher level language[46]. It is easy to see how, in principle, to deduce the transition matrix – simply take cells with various known states, put them in various neighborhoods, and see what they do. In the case of the other three things, it is progressively less obvious what to do.

Open problem #4: *Investigate multicellular development from this computer science point of view.*

4 Conclusion

Standard DNA lab techniques may be used to construct molecular Turing machines. Turing machines are simple but universal computers. This is a "nondeterministic" Turing machine with all the enormous parallelism that that implies – up to the point that we run out of DNA molecules. The known in vitro constructions will not be practical in competition with conventional computers, but the reasons for this trace to technological limitations that do not apply *in vivo*.

The view of "biology as a computer programming programming problem" may allow insight into biology and may also suggest new directions in which to extend computer science. For example, why is the biologically bizarre process of RNA editing there? From the computer science point of view we see that RNA editing is a very natural way to get Turing universal computational power, located in a very natural place in the biochemical information pathways. We have also given a natural sounding hypothesis about why junk DNA is there. As an example in the other direction, biology suggests a number of computational models whose mathematical properties are presently little known. Although computer scientists often in the past had been satisfied with proving some model (with an infinite tape) is Turing universal with some (unknown) polynomial slowdown ratio, or as powerful as a stack machine, and so on, it seems that biology may now for the first time be providing incentive to make finer distinctions than these.

5 Acknowledgment

Allan Schweitzer was a major contributor to this paper but wished not to be recorded as an "author."

6 References

Adleman, L.: Molecular computation of solutions to combinatorial problems, Science 266 (Nov 1994) 1021-1024.

von Ahsen, U., Davies, J., Schroeder, R.: Noncompetitive inhibition of group I intron RNA self-splicing

[45] It is also known that cells can change "state" (in our cellular automaton sense) as a function of time – they *do* have crude clocks.

[46] From the standpoint of biology, it is known that all of these capabilities exist, although they are not well understood at the moment, and it is not known if they are all actually used in the manner we have suggested. Thus "subroutine invocations" are sometimes known to biologists as "homeobox genes" and other times are called "genetic switches." Incidentally, we are using "higher level language" here in the computer science sense (e.g. MODULA, PROLOG) and we wish to disown any connection to Mantegna et al. 1994. There it was observed that introns in DNA have Zipf power law histograms, and various deep-sounding linguistic conclusions were drawn. Unfortunately Mantegna and his 6 coauthors were unaware that words of either fixed or variable lengths whose letters are drawn entirely at random from a fixed alphabet with fixed unequal letter probabilities, also exhibit Zipfian statistics.

by aminoglycoside antibiotics, Journal of Molecular Biology 226,4 (Aug 1992) 935-941.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., Watson, J.: Molecular biology of the cell (3rd ed.) Garland 1994.

Andre, P. and Dubois, O.: Using the expected number of solutions to optimize the resolution of a SAT instance, Compte Rendu de l'Académie des Sciences de Paris, tome 315, série I (1992) 217-220.

Baker, E.J.: 367-415 in Control of mRNA stability (eds. J.G. Belasco and G. Brawerman) Academic 1993.

Bartel, D.P., and Szostak, J.W.: Isolation of new ribozymes from a large pool of random sequences, Science 261 (1993) 1411-1417.

Benne, R., et al.: Major transcript of the frameshifted coxII gene from trypanosome mitochondria contains four nucleotides that are not encoded in the DNA, Cell 46 (1986) 819-826.

Benne, Rob: RNA editing in Trypanosomes, Molecular Biology Reports 16 (1992) 217-227; Eur. J. Biochem. 221 (1994) 9-23.

Berlekamp, Elwyn R., Conway, John H., Guy, Richard K.: Winning ways, for your mathematical plays, Academic Press, 1982.

Blum, B., Bakalara, N., Simpson, L.: A model for RNA editing..., Cell 60 (Jan 1990) 189-198.

Bonner, T.I., et al.: Reduction in the rate of DNA reassociation by sequence divergence, J. Molecular Biology 81 (1973) 123-135.

Brady, A.H.: The determination of Rado's noncomputable function Sigma(k) for four-state Turing machines, Math. Comput. 40, 62 (1983) 647-665.

Brail, L., Fan, E., Levin, D.B., Logan, D.M.: Improved polymerase fidelity in PCR-SSCPA, Mutation Research 303,4 (Dec 1993) 171-175.

Bray, D.: Protein molecules as computational elements in living cells, Nature 376 (1995) 307-312.

Britten, Roy J., & Davidson, Eric H.: Repetitive and non-repetitive DNA sequences and a speculation on the origins of evolutionary novelty, Quart. Rev. Biol. 46,2 (1971) 111-133.

Britten, R.J., and Kohne, D.E.: Repeated sequences in DNA, Science 161 (1968) 529-540

Brown, T.A. (ed): Essential Molecular Biology: a practical approach, IRL Press, Oxford, 1991.

Buro, M.: A contribution to the determination of Rado's $\Sigma(5)$, Diploma thesis, Aachen, 1990.

Cairns, J., Foster, P.L.,: Adaptive reversion of a frameshift mutation in Escherichia Coli, Genetics 128 (1991) 695-701; Mechanisms of directed mutation, Genetics 131 (1992) 783-789.

Cairns, J., Overbaugh, J, Miller, S.: The origin of mutants, Nature 335 (1988) 142-146.

Calderbank, A.R., Hammons, A.R. Jr., Kumar, P. V., Sloane, N.J.A., Solé, P.: The $\mathbf{Z}_4$-Linearity of Kerdock, Preparata, Goethals and Related Codes, IEEE Trans. Information Theory 40 (1994) 301-319

Thomas R. Cech: Chromosome end games, Science 266 (Oct 1994) 387-388.

Cech, Thomas R.: The chemistry of self-splicing RNA and RNA enzymes, Science 236 (1987) 1532-1539.

Daniel W. Celander and Thomas R. Cech: Visualizing the higher order folding of a catalytic RNA molecule, Science 251 (Jan 1991) 401-407.

Chakrabarti, S., Ranade, A., Yelick, K.: Randomized load balancing for tree structured computation, Proc. Scalable High Performance Computing Conf. (1994) 666-673.

Challberg, M.D. and Englund, P.T.: The effect of template secondary structure on vaccinia DNA polymerase, J. Biol. Chem. 254,16 (1979) 7820-7826.

Correll, R.A., Feagin, J.E., Riley, G.E., Strickland, T., et al.: Trypanosma brucei minicircles encode multiple guide RNAs which can direct editing of extensively overlapping sequences, Nucleic Acids Res. 21,18 (1993) 4313-4320.

Cotton, R.G.H.: Detection of single base changes in nucleic acids, Biochem.J. (1989) 1-10.

Crawford, J.and Auton, L.D.: Experimental Results on the Cross-Over Point in Satisfiability Problems, Proc. 11th Nat'l Conf. on Artificial Intelligence (AAAI-93) 21-27.

Crowder, H., and Padberg, M.W.: Solving large scale symmetric traveling salesman problems to optimality, Management Sci. 26 (1980) 495-509.

Darnell, James, Lodish, Harvey, and Baltimore, David: Molecular cell biology, Scientific American Books, New York, W.H. Freeman 1990.

Deininger, P.L., and Daniels, G.R.: The recent evolution of mammalian repetitive segments, Trends Gen. 2 (1986) 76-80.

Dewdney, A.K.: Computer recreations column, Scientific American (April 1985) 20-30. A hardware implementation of a Turing machine by G.Uhing found $B_5 \geq 1915$.

Dewdney, A.K.: The Turing omnibus, 61 excursions in computer science, Computer Science Press 1989.

Second DIMACS implementation challenge, a book planned to be published by American Mathematical Society, Providence, R.I.

Drake, J.W.: A constant rate of spontaneous mutation in DNA-based microbes, Proc. Nat'l. Acad. Sci. USA 88,16 (Aug 1991) 7160-7164.

Drake, J.W.: Rates of spontaneous mutation among RNA viruses, Proc. Nat'l. Acad. Sci. USA 90,9 (May 1993) 4171-4175

Driver, C.J., McKechnie, S.W.: Transposable elements as a factor in the aging of Drosophila melanogaster. Annals of the New York Academy of Sciences 673 (1992) 83-91.

Dubois, O., Andre, P., Boufhkad, Y., Carlier J.: Manuscript on SAT algorithm implementations. Should appear in the DIMACS challenge book (qv).

Edge, M.D., Greene, A.R., Heathcliff, G.R., Meacock, P.A., Schuch, W., et al.: Total synthesis of human leukocyte interferon gene, Nature 292 (1981) 756-763.

Ellington, A.D. and Szostak, J.W.: In vitro selection of RNA molecules that bind specific ligands, Nature 346 (1990) 818-822.

A. R. Fersht: Enzymatic editing mechanisms and the genetic code, Proc. Royal Soc. London B 212 (1981) 351-379.

Alan R. Fersht and J.W. Knill-Jones: Fidelity of replication of bacteriophage ϕX174 in vitro and in vivo, J. Molec. Biol. (1983) 633-654.

Fodor, S.P.A., Rava, R.P., Huang, X.C., Pease, A.C., Holmes, C.P., Adams, C.L.: Multiplexed biochemical assays with biological chips, Nature 364 (Aug 1993) 555-556.

Ford, E. and Ares, M. Jr.: Synthesis of circular RNA in bacteria and yeast using RNA cyclase ribozymes derived from a group I intron of phage T4. Proc. Nat'l Academy of Sciences USA 91,8 (Apr 1994) 3117-3121.

Foster, P.L., & Trimarchi, J.M.: Adaptive reversion of a frameshift mutation in E.Coli by simple base deletions in homopolymeric runs, Science 265 (1994) 407-409

Fraga, C.G., Shigenaga, M.K., et al.: Oxidative damage to DNA during aging: 8-hydroxy-2'-deoxyguanine in rat organ DNA and urine, Proc. Nat'l. Acad. Sci. USA 87 (1990) 4533-4537.

Fredricksen, H.: A survey of full length nonlinear shift register cycle algorithms, SIAM Review 24,2 (1982) 195-221

Garey, Michael R., and Johnson, David S.: Computers and intractability : a guide to the theory of NP-completeness, San Francisco : W. H. Freeman, 1979.

Gilbert, W.: Why genes in pieces, Nature (News and Views) 271 (1978) 501.

Göringer, H.U., Koslowsky, D.J., Morales, T.H., Stuart, K.: The formation of mitochondrial ribonucleoprotein complexes involving guide RNA molecules in Trypanosoma Brucei, Proc. Nat'l. Acad. Sci. USA 91 (Mar 1994) 1776-1780.

Green, N. Michael: Avidin, Adv. Protein Chem. 29 (1975) 85-133.

Greeve, J, Navaratnam, N, Scott, J.: Characterization of the apolipoprotein B mRNA editing enzyme: no similarity to the proposed mechanism of RNA editing in kinetoplastid protozoa, Nucleic Acids Research, 19,13 (1991) 3569-3576.

Guerrier-Takada, C., et al.: The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme, Cell 35 (1983) 849-857.

Hagerman, Paul J.: Flexibility of DNA, Ann. Rev. Biophys. Chem. 17 (1988) 265-288

Halford, S.E., Johnson, N.P.: The *Eco*RI restriction endonuclease with bacteriophage λ DNA, Biochem. J. 191 (1980) 593-604. See also Clore, Gronenborn, Davies: J.Molec.Biol. 155 (1982) 447-466.

Hjelmfelt, A., Weinberger, E.D., and Ross, J.: Chemical implementation of neural networks and Turing machines, Proceeding Nat'l Acad. Sci. USA 88,24 (1991) 10983-10987.

Inoue,T. and Orgel, L.E.: A nonenzymatic RNA polymerase model, Science 219 (1983) 859-862.

Jerne, Niels K.: The generative grammar of the immune system, Nobel Lecture 8 December 1984.

Joyce, G.: RNA evolution and the origins of life, Nature 338 (1989) 217-224.

Joyce, G.: Directed molecular evolution, Scientific American (Dec 1992) 90-97.

M.Kálmán et al.: Synthesis of a gene for human serum albumin and its expression in Saccharomyces Cerevisiae, Nucleic Acids Research 18,20 (1990) 6075-6081.

Karp, R.M., and Zhang, Y.: Randomized parallel algorithms for backtrack search and branch-and-bound computation, J. Assoc. Comput. Machin. 40 (1993) 765-789.

Khorana, H.G.: Harvey Lectures 62 (1968) 79. Pure and Appl. Chem 17 (1968) 349. The Biochem. J. 109 (1968) 709.

Kornberg, Arthur, and Baker, Tania A.: DNA Replication, 2nd ed. Freeman 1991.

Kricker, M.C., Drake, J.W., Radman, M.: Duplication-targeted DNA methylation and mutagenesis in the evolution of eukaryotic chromosomes, Proc. Nat'l Acad. Sci. USA, 89,3 (1992) 1075-1079.

Kruger, K., et al.: Self splicing RNA: Autoexcision and autocyclization of the ribosomal RNA intervening sequence of Tetrahymena, Cell 31 (1982) 147-157.

Kumar, R. & Levings, C.S. III: RNA editing of a chimeric maize mitochondrial gene transcript is sequence specific, Curr. Genet. 23 (1993) 154-159.

Kunkel, T.A., and Eckert: Polymerase chain reaction, p5-10 in: Current Communications in Molecular Biology; CSH Lab Press, Cold Spring Harbor NY (1989)

Lahue, R.S., Au, K.G., Modrich, P.: DNA mismatch correction in a defined system. Science, 245 (Jul 1989) 160-164.

Landweber, L.F., and Gilbert, W.: RNA editing as a source of genetic variation, Nature 363 (1993) 179-182.

Lewin, B.: Genes V, Oxford University Press 1994.

Lindahl, Tomas: Instability and decay of the primary structure of DNA, Nature 362 (Apr 1993) 709-715

Lipton, R.: Speeding up computations via molecular biology, Manuscript, Dec 9 1994, /ftp/pub/people/rjl/bio.ps on ftp.cs.princeton.edu.

Lombardi, D., Soldati, T., et al.: Rab9 functions in transport between late endosomes and the trans Golgi network, Euro. Molec. Biol. Org. Journal 12 (1993) 677-682.

Lonergan, K.M., and Gray, M.W.: Editing and transfer RNAs in Acantha amoeba Castellanii Mitochondria, Science 259 (1993) 812-818.

Lorch, J.R., and Szostak, J.W.: In vitro evolution of new ribozymes with polynucleotide kinase activity, Nature 371 (Sep 1994) 31-36.

Lundberg, K.S., Shoemaker, D.D., et al.: High-fidelity amplification using a thermostable DNA polymerase isolated from Pyrococcus furiosus. Gene 108,1 (Dec 1991) 1-6.

Maniatis, Tom, and Reed, Robin: The role of small nuclear ribonucleoprotein particles in pre-mRNA splicing, Nature 325 (1987) 673-678.

Mantegna, R.N. et al.: Linguistic features of noncoding DNA sequences, Phys. Rev. Lett. 73,23 (Dec 1994) 3169-3172.

Markham, A.F., et al.: Solid phase phosphotriester synthesis of large oligodeoxyribonucleotides on a polyamide support, Nucleic Acids Research 8,22 (1980) 5193-5205.

Maslov, Dimitri A., Simpson, Larry: The polarity of editing within a multiple gRNA mediated domain is due to formation of anchors for upstream gRNAs by downstream editing, Cell 70 (1992) 459-467.

Maslov, D.A., et al.: Evolution of RNA editing in kinetoplastid protozoa, Nature 368 (Mar 1994) 345-348.

Mathews, Christopher K. & van Holde, K.E.: Biochemistry, Benjamin-Cummings 1990.

Mathews, C.K., et al.: Bacteriophage T4, Amer. Soc. Microbiologists 1983.

McGinnis, William, and Kuziora, Michael: The molecular architects of body design, Scientific American (Feb 1994) 58-66.

Michel, P.: Busy beaver competition and Collatz-like problems, Arch. Math. Logic 32 (1993) 351-367.

Miller, Stanley L. & Orgel, Leslie E.: The origins of life on the earth, Englewood Cliffs, N.J., Prentice-Hall 1974.

Minsky, Marvin Lee: Computation: finite and infinite machines, Englewood Cliffs, N.J., Prentice-Hall 1967.

Mitchell, D., Selman, B. and Levesque, H.J.: Hard and easy distributions of SAT problems. Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), San Jose, CA (July 1992) 459-465.

Mitchison, G.J., and Wilcox, Michael: Rules governing cell division in Anabaena, Nature 239 (1972) 110-111.

Miyoshi, K-I. et al: Solid phase synthesis of polynucleotides IV, Nucleic Acids Research 8,22 (1980) 5507-5517.

Monien, B. and Speckenmeyer, E.: Solving satisfiability in less than 2^n steps, Discrete Applied Mathematics, 10 (1985) 287-295.

Mueller, M.W., Hetzer, M., Schweyen, R.J.: Group II intron RNA catalysis of progressive nucleotide insertion: a model for RNA editing. Science 261 (Aug 1993) 1035-1038.

Neumann, J. von, and Burks, A.W.: Theory of self reproducing automata, U. Illinois Press 1966.

New England Biolabs catalog (32 Tozer Road, Beverly, MA 01915-5599, USA.)

Nielsen, P., Egholm, M., et al.: Sequence-selective recognition of DNA by strand displacement with a thymine substituted polyamide, Science (Dec 1991) 1497-1500.

Orgel, Leslie E.: The origins of life: molecules and natural selection, Wiley 1973.

Orgel, L.E.: The origin of life on the earth, Scientific American (Oct 1994) 77-83.

Orgel, L.E., and Crick, F.H.C.: Selfish DNA: the ultimate parasite, Nature 284 (1988) 604-607.

Pace, N.R.: Origin of life – facing up to physical setting, Cell 65 (1991) 531-533

Padberg, M.; Rinaldi, G. Optimization of a 532-city symmetric traveling salesman problem by branch and cut, Operations Research Letters 6,1 (March 1987) 1-7.

Patthy, László: Introns and exons, Current Opinion in structural biology 4 (1994) 383-392.

Pawson, T: Protein molecules and signalling networks, Nature 373 (1995) 573-580.

Peebles, C.L., Perlman, P.S., et al.: A self-splicing RNA excises an intron lariat, Cell 44,2 (Jan 1986) 213-223.

Prescott, D.M.: The unusual organization and processing of genomic DNA in hypotrichous cilates, Trends in Genetics 8,12 (Dec 1992) 439-445.

Prescott, D.M.: Cutting, Splicing, Reorderding, and elimination of DNA sequences in hypotrichous ciliates, BioEssays 14,5 (May 1992) 317-324.

PROMEGA Biological research products catalog, Promega corp. 2800 Woods hollow road, Madison WI 53711-5399.

Ptashne, Mark: A genetic switch, 2nd ed, Cell Press and Blackwell Scientific Publications 1992.

Roberts, R.J.: Restriction and modification enzymes and their recognition sequences, Gene 4,3 (1978) 183-194; Nuc. Acids Res. 16 suppl (1988) r271-r313. Also available through BIONET as <ROBERTS>RESTRICT.{NAR, DOC}.

Robinson, R.M.: Minsky's small universal Turing machine, Int'l J. Math 2,5 (1991) 551-562.

Ruth, J.: Oligonucleotide with reporter groups attached to the base, chapter 11 in Oligonucleotides and analogs: a practical approach, (F. Eckstein ed.) IRL Press/Oxford U Press, 1991.

Saiki, R.K., et al.: Primer directed enzymatic amplification of DNA with thermostable DNA polymerase, Science 239 (1988) 487-491.

Sambrook, J., Fritsch, E.F., and Maniatis, T. Molecular cloning : a laboratory manual (2nd. ed.) Cold Spring Harbor, N.Y. : Cold Spring Harbor Laboratory, 1989.

Sassanfar, M. and Szostak, J.W.: An RNA motif that binds ATP, Nature 364 (Aug 1993) 550-553

Saul, R.L., Ames, B.N.: Background levels of DNA damage in the population, Basic Life Sciences 38 (1986) 529-535.

Seiwert, S.D., and Stuart, K.: RNA editing: transfer of genetic information from gRNA to precursor mRNA in vitro, Science 266 (Oct 1994) 114-116.

Selman, B.; Levesque, H.; Mitchell, D. A new method for solving hard satisfiability problems. Pages 440-446 in: AAAI-92. Proceedings Tenth National Conference on Artificial Intelligence. (AAAI-92. Proceedings Tenth National Conference on Artificial Intelligence, San Jose, CA, USA, 12-16 July 1992). Menlo Park, CA, AAAI Press, 1992.

Sharp, Philip A.: Splicing of messenger RNA precursors, Science 235 (1987) 766-771.

Shore, D., Langowski, J., Baldwin, R.L.: DNA flexibility studied by covalent closure of short fragments into circles, Proc. Nat'l Adcad. Sci. USA 78,8 (1981) 4833-4837

Larry Simpson: RNA editing – a novel genetic phenomenon?, Science 250 (Oct 1990) 512-513.

Larry Simpson and Dmitri A. Maslov: RNA editing and the evolution of parasites, Science 264 (Jun 1994) 1870-1871.

Simpson, Larry, and Shaw, Janet: RNA editing and the mitochondrial cryptogens of kinetoplastid protozoa, Cell 57 (May 1989) 355-366.

Sogin,M. et al.: Phylogenetic meaning of the kingdom concept, an unusual ribosomal DNA from *Giardia lamblia*, Science 243 (1989) 75-77.

Stryer, Lubert: Biochemistry (3rd ed.), Freeman 1988.

Stuart, K.: RNA editing in trypanosomatid mitochondria, Annual Rev. Microbiol. 45 (1991) 327-344.

Tausta, S.L., Turner, L.R., Buckley, L.K., Klobutcher, L.A.: High fidelity developmental excision of Tec1 transposons and internal eliminated sequences in Euplotes crassus, Nucleic Acids Res. 19,12 (1991) 3229-3236

Terry, B.J., Jack, W.E., Rubin,R.A., Modrich,P.: Thermodynamic parameters governing interaction of EcoRI endonuclease with specific and nonspecific DNA sequences, J.Biol.Chem. 258,16 (1983) 9820-9825

Tonegawa, Susumu: Somatic generation of immune diversity, Nobel lecture, December 8 1987.

Tsagris, M, Tabler, M., Sanger, H.L.: Ribonuclease T1 generates circular RNA molecules from viroid-specific RNA transcripts by cleavage and intramolecular ligation, Nucleic Acids Research 19,7 (Apr 1991) 1605-1612.

Uhlenbeck, O.C.: A small catalytic oligonucleotide, Nature 328 (1987) 596-600.

J. Watson, Hopkins, Roberts, Steitz, Weiner: Molecular biology of the gene (4th ed), Benjamin-Cummings 1987

B. Weiss et al: Enzymatic breakage and joining of DNA VI, J.Biol.Chem. 243 (1968) 4543.

Wetmur, J.G.: Hybridization and renaturation kinetics of nucleic acids, Ann. Review of Biophysics and Biophysical Chemistry 5 (1976) 337-361.

J.G. Wetmur and N. Davidson: Kinetics of renaturation of DNA, J.Molec. Biol. (1968) 31, 349-370.

Woodruff, R.C: Transposable DNA elements and life history traits. I. Transposition of P DNA elements in somatic cells reduces the lifespan of Drosophila melanogaster. Genetica 86(1-3) (1992) 143-154.

7 Appendix A: Finite state and Turing machine terminology

A *finite state machine* is a finite set of states and finite list of *transition rules.* Each transition rule is of the form "if you are in state '52' and encounter an input 'c', then go to state '31' and output 'g'." The inputs and outputs are optional.

A *Turing machine* (TM) is just like a finite state machine, except that it is equipped with a 1-dimensional "tape" that can be used for storing information. The tape is infinite and is divided into squares, and in each square, a character may be written. The machine is defined by a finite set of rules of the form "if you are in state '76' and the current tape square has the character 'd', then go to state '23', overwrite the current tape square with 'h', and move one square 'left' on the tape." In these rules, the overwriting is optional, and the movement need not be leftward, it could instead be rightward, or no movement at all, depending on the rule.

By convention, Turing machines usually have a special 'halt' state from which no transition is possible, and initially, all but a finite portion of the tape squares are blank. Once the initial message (or "input," or "data and program") has been pre-written on the tape, and you turn on the Turing machine, the rest of its actions are predetermined and no further interaction with a human being is required.

A computer is said to have *universal* computational power over some class $\mathcal{C}$ if it can be programmed to emulate any other kind of computer in $\mathcal{C}$.

Turing machines are universal over the class of *all* machines that would commonly be called computers. Furthermore, there exist particular universal Turing machines (several have been constructed) and any one of them can emulate any other reasonable kind of computer with at most polynomial slowdown. (Really, the present claims are not clearly defined since we haven't said what we mean by "computer," but there are precise emulation theorems involving very large classes of computational devices.) For this reason (loosely speaking) problems insoluble by Turing machines are called Turing-"Undecidable."

A so-called *nondeterministic* Turing machine (NDTM) has a set of transition rules which need not specify a unique action – they can specify two or more contradictory actions. For example, one rule could say to move right, the other could say to move left. In that case, the NDTM splits into two NDTMs, one of which moves left and one of which moves right, and they both continue on from there.

We now speak informally to save time: The class of problems which a universal Turing machine can solve in a number of steps bounded by a polynomial of the number of bits in its input, is called "P." The class of problems which a nondeterministic universal Turing machine can solve (that is, which at least *one* of the alter egos solves) in a number of steps bounded by a polynomial of the number of bits in its input, is called "NP." A problem class X, such that any problem in NP can be rephrased as a problem of the form X (and the rephrasing task is in P) is called "NP-complete." The traveling salesman problem, graph coloring, and a logic problem called "SAT" are known NP-complete problem classes. It is widely conjectured that P $\subset$ NP and that NP-complete problems would require, in the worst case, exponential time to be solved by a (deterministic) TM.

PSPACE is the class of problems which can be solved by TMs which only visit a region of their tape having polynomially large extent. EXPTIME is the class of problems which can be solved by TMs in exponential time. Solving positions of certain games, such as $n \times n$ checkers, is known to be EXPTIME complete. It is widely conjectured that NP $\subset$ EXPTIME, so that even NDTMs could not solve such problems in polynomial time. However, a TM which not only could proliferate into several children (ala an NDTM) but also could get information *back* from said children, *could* solve $n \times n$ checkers in polynomial time. This fact illustrates the value of interprocessor communication.

8 Appendix B: Modified DeBruijn sequences

The maximum possible number of letters ℓ in a circular DNA string, so that no k-letter substring occurs twice, is $\ell = A^k$ where $A > 1$ is the alphabet size and $A = 4$ for DNA. Here $\ell \le A^k$ is obvious, and this bound is met by radix-A DeBruijn sequences, see Frederickson 1982. For noncircular DNA, $\ell = A^k + k - 1$, achieved by noncircular DeBruijn sequences. If we also disallow the appearance of any reversed order and complemented k-letter substring ("modified DeBruijn problem"), then $\ell \le (A^k + A^{\lfloor k/2 \rfloor})/2 + k - 1$. Conjecturally, this bound is tight when $k = 1$, when $(k, A) = (4, 2)$, when $k = 2$, and for no other (k, A). (The first two facts are easily proven. We have proven the third claim when $2 \le A \le 14$ but, suprisingly, we have so far been unable to prove it for all A. The fourth claim is almost wholy conjectural.) Also, conjecturally the upper bound is asymptotically tight when $A \to \infty$ with k fixed (indeed when $k = 3$ and A is even, the formula $\ell = A^3/2 + 2$ seems always to hold!). The usual nonconstructive randomizing argument shows that $\ell \ge k + A^{k/2}$. The best lower bounds that we know for the length ℓ of modified DeBruijn sequences for various (k, A) are given in the table below ("`.`" means "meets the upper bound formula" and "`!`" means "proven optimal by exhaustive search [by Henry Cejtin].").

.	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=11	k=12
A=2	1.	4.	6!	13.	20!	39!	70!	137	264	441	1034	1494
A=3	2.	7.	16!	45	117	315	811	2078	5032	12377		
A=4	2.	11.	34!	133	516	1749	8198					
A=5	3.	16.	64	300	1359	5701						
A=6	3.	22.	110	632	3892	18307						
A=7	4.	29.	172	1106	6869							
A=8	4.	37.	258	1946	16388							
A=9	5.	46.	360	3019	21831							
A=10	5.	56.	502	4622								
A=11	6.	67.	656	6636								
A=12	6.	79.	866	9496								
A=13	7.	92.	1067	12650								
A=14	7.	106.	1374									

An example of a modified DeBruijn sequence with $A = 4$, $k = 4$, and $\ell = 133$ is

```
abcdabdcccaadddbcdbcacadbaacbccaba
bbdadabbadbcbcbbaabdaabbbbccdbddbd
acbdbdcaabcbaadadbbdbadcdddcdccacd
cbbcdcabddabacbaccbbbddddaadcba.
```

9 Appendix C: DNA basics and terminology

At elevated temperature (≈ 90 C) the two strands which comprise a double-stranded (dsDNA) DNA molecule will dissociate or "denature." The resulting single-stranded DNA (ssDNA) is a linear polymer consisting of a sequence of chemical subunits joined in sequence by strong (covalent) bonds. Each subunit, or "nucleotide," consists of a constant part, a phosphorylated, cyclic, 5-carbon sugar (ribose), attached to a variable part, which is one of four biochemical bases: the purines adenine (A) and guanine (G) or the pyrimidines thymine (T) and cytosine (C). [47]. ssDNA strands have polarity. The monomers have functional groups at two sugar sites, designated $3'$ and $5'$, and during polymerization (below) the monomers are linked together $5' \rightarrow 3'$ in parallel orientation. Ignoring chemical modification to the bases [48] the ssDNA molecule is completely specified by the sequence of its nucleotide bases[49].

While each DNA strand is held together by strong bonds, in solution weak (hydrogen) bonds act to cause *specific* base-pairs to associate: A with T and G with C: these base pairs are "complementary." For any ssDNA sequence there is a complementary sequence consisting of complementary bases written in the opposite order: e.g. the sequence ATTCGCT is complementary to AGCGAAT. The sequences of the two ssDNA strands comprising a piece of dsDNA are complementary, and the strands are held together by the hydrogen bonding between each complementary base. Because it is difficult to seperate two bases without breaking the bonds of their neighbors, the dsDNA binding is highly "cooperative" and the "melting" (dissociation) occurs at a defined temperature[50]. When the temperature of the melt is reduced, complementary ssDNA strands re-bind, or "anneal," to form the familiar dsDNA form of DNA.

The distinctive chemical sub-groups of each base which participate in hydrogen-bonding of complementary ssDNA sequences to form dsDNA are buried near the axis of the molecule and not "visible" to the outside. Each base, however, has other distinctive sub-groups by which it can be identified, and these groups are chemically accessible to the exterior of the DNA molecule via the so-called "major" and "minor" groves. A large class of "DNA binding" proteins interact with these exposed base functional groups and bind to specific sequences of dsDNA. Among this class of proteins are the so-called "restriction endonucleases," proteins which bind to specific sub-sequences ("recognition sites") of dsDNA and then cut the covalent bonds linking the individual ssDNA at specific sites relative to the location of the binding sequence.

The covalently-bonded "backbone" of ssDNA, neglecting the bases, is a linear sequence of identical subunits

5'-[P-S]-[P-S]-[P-S]- ... [P-S]- 3'

where [P-S] denotes a (nucleoside monophosphate) monomer consisting of a sugar unit with one attached phosphate group. During polymerization, the reaction between the acidic phosphate group of the incoming monomer and the 3' terminal hydroxyl group of the growing strand is a dehydration reaction much like the reaction of an acid and an alcohol to form an ester plus water. The polymerization of nucleoside monophosphates is, however, thermodynamically unstable with respect to the inverse reaction, hydrolysis: this fact has important implications for the stability of any information-representation involving DNA, discussed below. The polymerization of DNA can be made thermodynamically favorable by choosing monomers consisting of a sugar group attached to three phosphate groups. In the reaction of these nucleoside triphosphate monomers (technically "nucleotides") to form DNA, two of the phosphate groups are split off to produce pyrophosphate $HOP_2O_6^{3-}$. This secondary reaction is sufficiently favored by thermodynamics that the "excess" free energy can be used to "drive" the unfavorable polymerization reaction. A great many reactions in biology, similarly unfavorable, are "driven" by coupling to other reactions, very often the hydrolysis of the nucleoside triphosphates ATP or GTP. For the case of DNA polymerization, however, the monomers bring, in effect, the required excess free energy to the reaction, and it is not necessary to couple the reaction to any reaction.

Enzymes known as DNA polymerases catalyze the "template-directed" polymerization of DNA. Provided with an adequate supply of nucleotide triphosphates and ssDNA "template" strands, polymerase binds one of the ssDNA templates and proceeds to "copy" the template to form a complementary strand (which can bind to the template to form dsDNA). Polymerization of the growing strand proceeds by the addition of free nucleotides to the 3' end of the growing strand. As the polymerase can only extend an existing strand, and not initiate a new strand, a short ssDNA "primer" complementary to a sub-sequence of the template must be added to initate polymerization. Provided with a sufficient supply of primers, cycles of polymerization and denaturation can be used to exponentially "amplify" a population of ssDNA, a tech-

[47] A note on nomenclature: These are the names of the "bare" bases. Attachment to an un-phosphorylated sugar produces the *nucleosides* adenosine, guanosine, thymidine, and cytidine One, two, or three phosphate groups can be attached to the nucleosides to produce the *nucleotide mono-,di- and tri-phosphates*: *eg* adenosine 5'-monophosphate. The monophosphate nucleotide forms occuring in DNA are properly referred to as: adenylate, gunaylate, thymidylate, and cytidylate.

[48] Modifications, e.g. methylation, of specific bases to protect DNA sequences from degredation by restriction enzymes, occurs frequently *in vivo*.

[49] Not quite: it is important to keep track of the state of the $3'$ and $5'$ ends, to which either a hydroxyl or phosphate group may be attached, as well. In special cases, it is even possible to attach a $3'$ terminal nucleotide lacking the $3'$ hydroxyl, which blocks further polymerization.

[50] The melting temperature depends upon sequence, because G-C base-pairing, involving three hydrogen bonds, is tighter than the A-T interaction.

nique know as "polymerase chain reaction" or PCR.

The phosphodiester bond is vulnerable to hydrolysis. When hydrolysis breaks ssDNA strands, a "nick" in dsDNA is repairable by the enzyme ligase, provided that the phosphate group remains attached to the 5' sugar. Because the reaction is thermodynamically unfavorable, however, ligase couples the repair reaction to the hydrolysis of exogenous ATP.

Machine-synthesis of ssDNA does not utilize polymerase and requires very different reagents and reactions than occur *in vivo.* In 1980 the yields for each such nucleotide addition (Miyoshi 1980, Markham 1980) were in the 60-95% range, but things have improved since then and $> 99\%$ is now available, enabling the synthesis of oligonucleotide ssDNA sequences with lengths of over 100 nucleotides. The cost of made-to-order DNA is about $2 per base. Methods based on *ligating fragments* which were synthesized in ssDNA form, then converted to dsDNA via polymerase, amplified in bacteria, and selected for correctness, have been used to make a dsDNA 1761 base pairs long, which is the current record (Kalman, 1990). The initial fragments, after cloning, had a $\approx$0.5% error rate per base.

Techniques for the attachment of DNA to magnetic beads, etc., are based on the properties of the bacterial protein streptavidin, which tightly binds to the molecule biotin, a common biochemical cofactor. Biotin can be covalently linked to a nucleotide base, usually via a short "coupler," while streptavidin can be similarly attached to a substrate or bead; the affinity of streptavidin for biotin then causes the DNA (ss or ds) to bind to the bead. Photosensitive coupling systems are available which debind on exposure to light.