JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 5, Number 4, 1998
Mary Ann Liebert, Inc.
Pp. 615–629

A Sticker-Based Model for DNA Computation

SAM ROWEIS,[1,2] ERIK WINFREE,[1,2] RICHARD BURGOYNE,[1,3] NICKOLAS V. CHELYAPOV,[1,4] MYRON F. GOODMAN,[1,5] PAUL W. K. ROTHEMUND,[1,4] and LEONARD M. ADLEMAN[1,4]

ABSTRACT

We introduce a new model of molecular computation that we call the *sticker model*. Like many previous proposals it makes use of DNA strands as the physical substrate in which information is represented and of separation by hybridization as a central mechanism. However, unlike previous models, the stickers model has a random access memory that requires no strand extension and uses no enzymes; also (at least in theory), its materials are reusable. The paper describes computation under the stickers model and discusses possible means for physically implementing each operation. Finally, we go on to propose a specific machine architecture for implementing the stickers model as a microprocessor-controlled parallel robotic workstation.

In the course of this development a number of previous general concerns about molecular computation (Smith, 1996; Hartmanis, 1995; Linial *et al.*, 1995) are addressed. First, it is clear that general-purpose algorithms can be implemented by DNA-based computers, potentially solving a wide class of search problems. Second, we find that there are challenging problems, for which only modest volumes of DNA should suffice. Third, we demonstrate that the formation and breaking of covalent bonds is not intrinsic to DNA-based computation. Fourth, we show that a single essential biotechnology, sequence-specific separation, suffices for constructing a general-purpose molecular computer. Concerns about errors in this separation operation and means to reduce them are addressed elsewhere (Karp *et al.*, 1995; Roweis and Winfree, 1999). Despite these encouraging theoretical advances, we emphasize that substantial engineering challenges remain at almost all stages and that the ultimate success or failure of DNA computing will certainly depend on whether these challenges can be met in laboratory investigations.

Key words: molecular computation, DNA computation, sticker model.

1. INTRODUCTION

MUCH OF THE RECENT INTEREST IN MOLECULAR COMPUTATION has been fueled by the hope that it might some day provide the means for constructing a massively parallel computational platform capable of attacking problems which have been resistant to solution with conventional architectures. Model architectures

[1]Laboratory for Molecular Science, University of Southern California, Los Angeles, California.
[2]Computation and Neural Systems Option, California Institute of Technology, Pasadena, California.
[3]Department of Biomedical Engineering, University of Southern California, Los Angeles, California.
[4]Department of Computer Science, University of Southern California, Los Angeles, California.
[5]Department of Biological Sciences, University of Southern California, Los Angeles, California.

have been proposed which suggest that DNA based computers may be flexible enough to tackle a wide range of problems (Adleman, 1994, 1996; Amos *et al.*, 1999; Lipton, 1995; Boneh *et al.*, 1996; Beaver, 1995; Rothemund, 1996), although fundamental issues such as the volumetric scale of materials and fidelity of various laboratory procedures remain largely unanswered.

In this paper we introduce a new model of molecular computation that we call the *sticker model*. Like many previous proposals, it makes use of DNA strands as the physical substrate in which information is represented and of separation by hybridization as a central mechanism. However, unlike previous models, the stickers model has a random access memory that requires no strand extension, uses no enzymes, and (at least in theory) its materials are reusable.

The paper begins by introducing a new way of representing information in DNA, followed by an abstract description of the basic operations possible under this representation. Possible means for physically implementing each operation are discussed. Finally, we go on to propose a specific machine architecture for implementing the stickers model as a microprocessor-controlled parallel robotic workstation, employing only technologies which exist today.

2. THE STICKERS MODEL

2.1. Representation of information

The stickers model employs two basic groups of single stranded DNA molecules in its representation of a bit string. Consider a *memory strand* N bases in length subdivided into K nonoverlapping regions each M bases long (thus, $N \geq MK$). Each region is identified with exactly one bit position (or equivalently one boolean variable) during the course of the computation. We also design K different *sticker strands* or simply *stickers*. Each sticker is M bases long and is complementary to one and only one of the K memory regions. If a sticker is annealed to its matching region on a given memory strand then the bit corresponding to that particular region is *on* for that strand. If no sticker is annealed to a region, then that region's bit is *off*. Figure 1 illustrates this representation scheme.

Each memory strand *along with* its annealed stickers (if any) represents one bit string. Such partial duplexes are called *memory complexes*. A large set of bit strings is represented by a large number of *identical* memory strands each of which has stickers annealed only at the required bit positions. We call such a collection of memory complexes a *tube*. This differs from previous representations of information using DNA in which the presence or absence of a particular subsequence in a strand corresponded to a particular bit being on or off (e.g., see Adleman, 1994; Lipton, 1995). In this new model, each possible bit string is represented by a unique *association* of memory strands and stickers whereas previously each bit string was represented by a unique molecule.

To give a feel for the numbers involved, a reasonable size problem (for example, breaking DES as discussed in Adleman *et al.*, 1999), might use memory strands of roughly 12,000 bases (N), which represent 580 binary variables (K) using 20 base regions (M).

The information density in this storage scheme is $(1/M)$ bits/base, directly comparable to the density of previous schemes (Adleman, 1994; Boneh *et al.*, 1996; Lipton, 1995). We remark that while information storage in DNA has a theoretical maximum value of 2 bits/base, exploiting such high values in a separation-based molecular computer would require the ability to reliably separate strands using only single base mismatches.

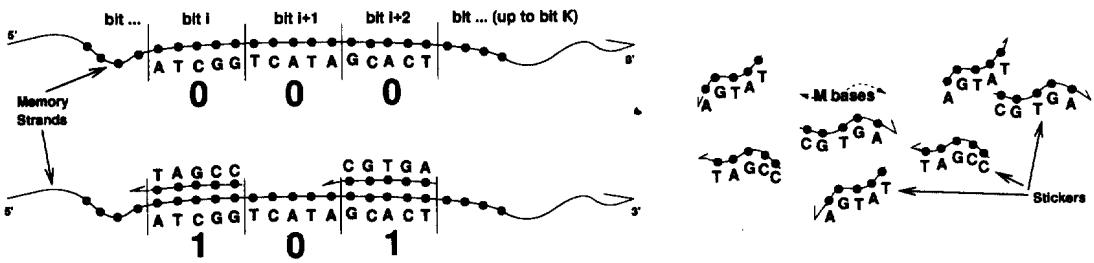

FIG. 1. A memory strand and associated stickers (together called a *memory complex*) represent a bit string. The top complex on the left has all three bits *off*; the bottom complex has two annealed stickers and thus two bits *on*.

Instead, we choose to sacrifice information density in order to make the experimental difficulties less severe.

2.2. *Operations on sets of strings*

We now introduce several possible operations on sets of bit strings which together turn out to be quite flexible for implementing general algorithms. The four principle operations are *combination* of two sets of strings into one new set, *separation* of one set of strings into two new sets, and *setting* or *clearing the* k^{th} *bit* of every string in a set. Each of these logical set operations has a corresponding interpretation in terms of the DNA representation introduced above. Figure 2 summarizes these required DNA interactions.

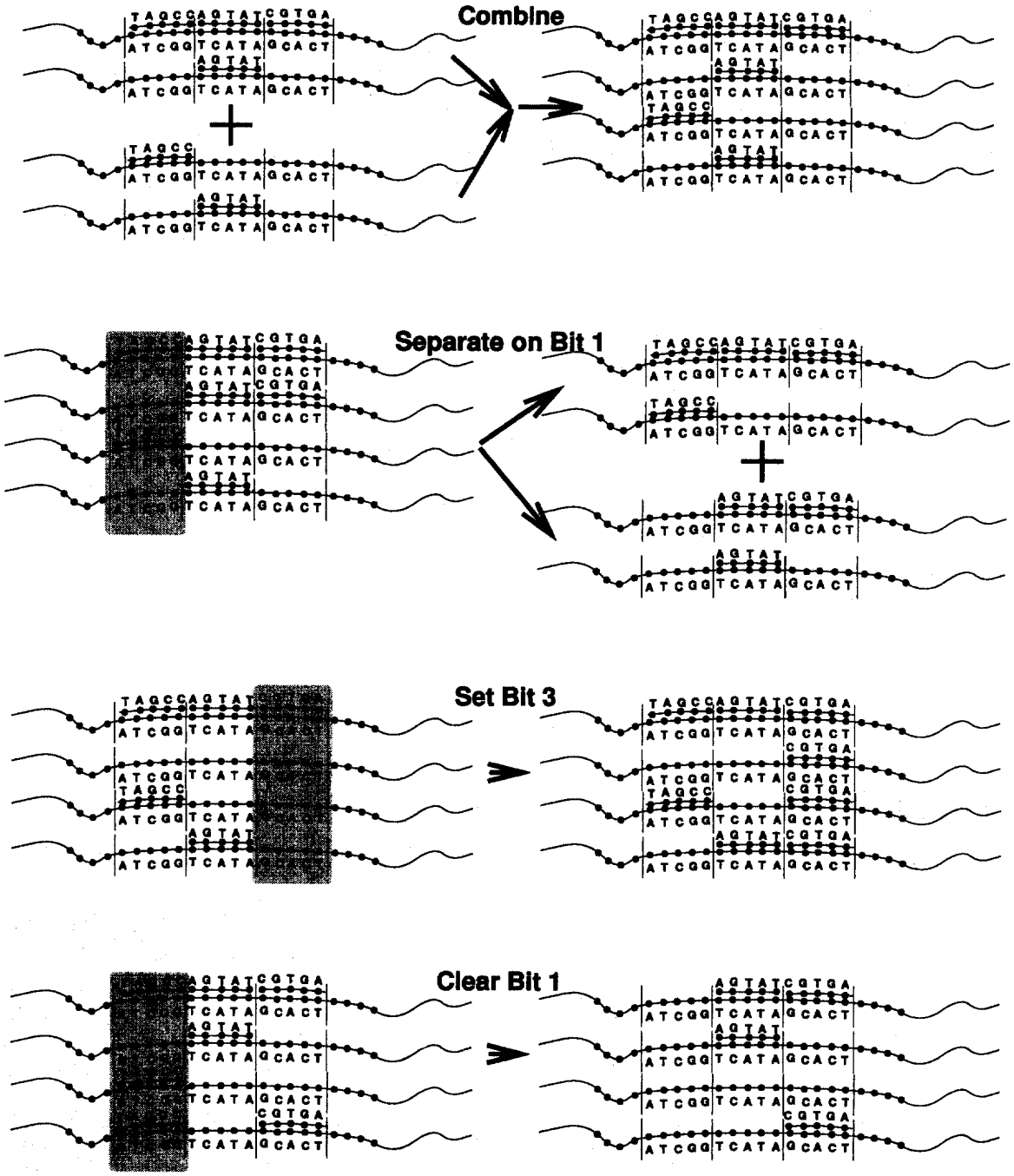

FIG. 2. DNA manipulations required for the four operations of the stickers model.

Instead, we choose to sacrifice information density in order to make the experimental difficulties less severe.

2.2. *Operations on sets of strings*

We now introduce several possible operations on sets of bit strings which together turn out to be quite flexible for implementing general algorithms. The four principle operations are *combination* of two sets of strings into one new set, *separation* of one set of strings into two new sets, and *setting* or *clearing the* k^{th} *bit* of every string in a set. Each of these logical set operations has a corresponding interpretation in terms of the DNA representation introduced above. Figure 2 summarizes these required DNA interactions.

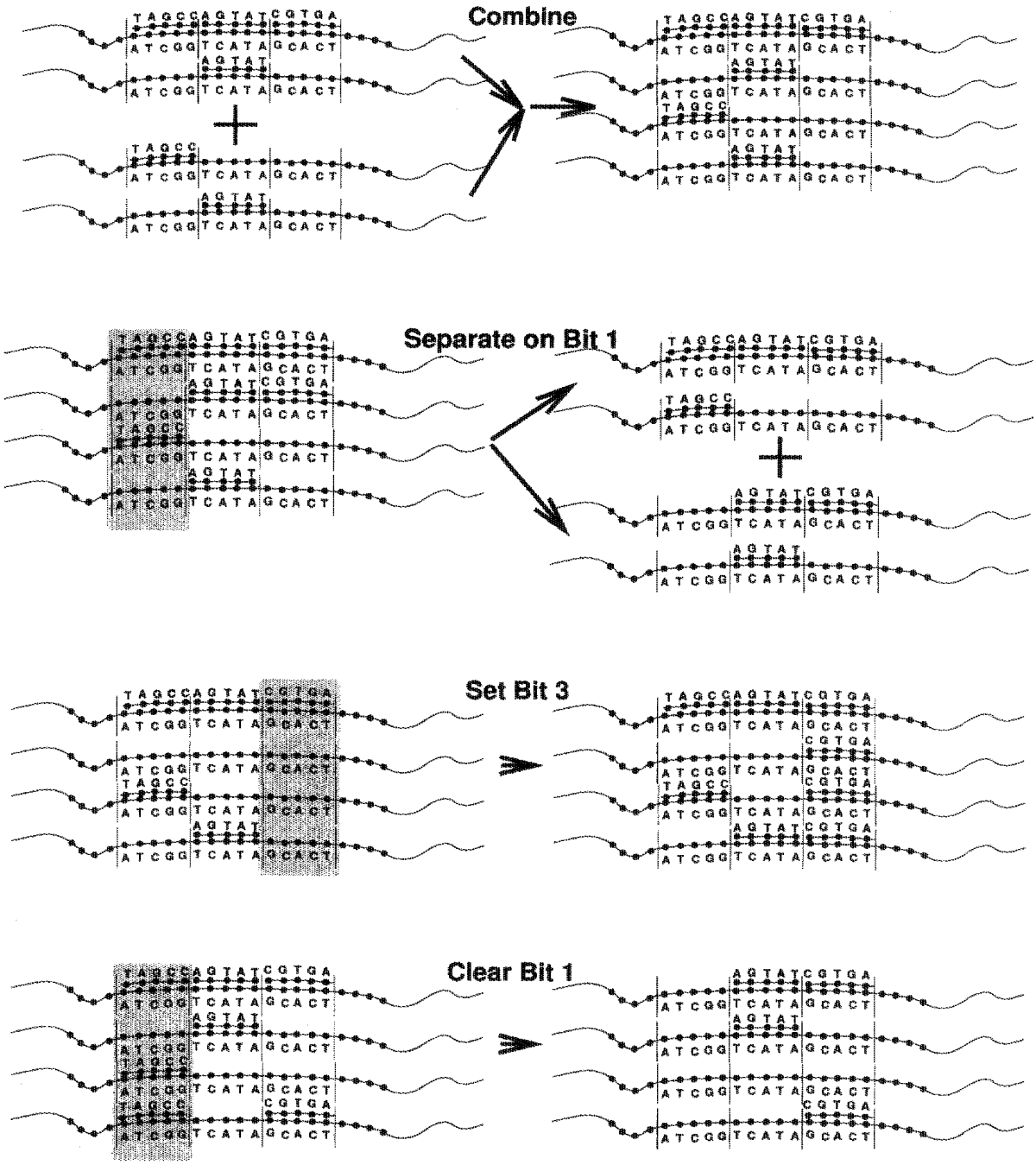

FIG. 2. DNA manipulations required for the four operations of the stickers model.

- The most basic operation is to *combine* two sets of bit strings into one. This produces a new set containing the multi-set union of all the strings in the two input sets. In DNA, this corresponds to producing a new tube containing all the memory complexes (with their annealed stickers undisturbed) from both input tubes.
- A set of strings may be *separated* into two new sets, one containing all the original strings having a particular bit *on* and the other all those with the bit *off*. This corresponds to isolating from the set's tube exactly those complexes with a sticker annealed to the given bit's region. The original input set (tube) is destroyed.
- To *set* (turn on) a particular bit in every string of a set, the sticker for that bit is annealed to the appropriate region on every complex in the set's tube (or left in place if already annealed).
- Finally, to *clear* (turn off) a bit in every string of a set, the sticker for that bit must be removed (if present) from every memory complex in the set's tube.

Computations in this model consist of a sequence of combination, separation, and bit setting/clearing operations. This sequence must begin with some initial set of bit strings and must ultimately produce one (possibly null) set of strings deemed to be "the answers." We call the tube containing the initial set of bit strings the *mother tube* for a computation. Thus, to complete our theoretical description of how to compute with the stickers model, we must describe how to create a mother tube of memory complexes and also how to read out at least one bit string from a (possibly empty) final tube of answers (or recognize that the tube contains no strands). We consider creation of the mother tube first:

- It will suffice for our purposes to consider creating a mother tube which corresponds to the (K, L) *library set* of strings. A (K, L) *library set* contains strings of length K generated by taking the set of all possible bit strings of length L followed by $K - L$ zeros. There are thus 2^L length K strings in the set.[1]

Our paradigm of computation will generally be to cast hard problems as large combinatorial searches over inputs of length L. We search for the few rare "answer" strings by processing all 2^L possible inputs in parallel and eliminating those that fail the search criteria. It is important that the memory strand we design may have more than L bit regions. The first L bits represent the encoding of the input and are the random portion of the initial library. The remaining $K - L$ bits are used for intermediate storage and answer encoding and are initially off on all complexes. All bits can be written to and read from later in the computation as needed. In this way creating a mother tube which is a (K, L) *library set* corresponds to generating all possible inputs (of length L) and zeroing the workspace (length $K - L$).

Lastly, we indicate how to obtain a solution at the end of the computation:

- To read a string from the final "answer" set, one memory complex must be isolated from the answer tube and its annealed stickers (if any) determined. Alternately, it must be reported that the answer tube contains no strands.

2.3. *Example problem*

To illustrate the power of the operations defined above we work through the solution of the NP-Complete[2] *Minimal Set Cover* problem (Garey and Johnson, 1979) within the stickers model. Informally, assume we are given a collection of B bags each containing some objects. The objects come in A types. The problem is to find the smallest subset of the bags which between them contain at least one object of every type. Formally the problem is as follows: *Given a collection* $C = \{C_1, \ldots, C_B\}$ *of subsets of* $\{1, \ldots, A\}$ *what is the smallest subset* I *of* $\{1, \ldots, B\}$ *such that* $\cup_{i \in I} C_i = \{1, \ldots, A\}$? The solution of the problem in our model is straightforward. We create memory complexes representing all possible 2^B choices of bags. We mark all those which include bag i as containing every type appearing in the subset C_i. Then we separate out those complexes which have been marked as containing all A types and read out the one(s) which uses the fewest bags. Formally, the sticker algorithm for minimal set cover is:

[1] For example, the (7, 3) *library* set is the set {0000000, 0010000, 0100000, 0110000, 1000000, 1010000, 1100000, 1110000}.

[2] Technically, the NP-Complete version of this problem is the binary decision version in which we ask if there exists a collection of a *particular* size that covers the set, not for the collection of the smallest size.

```
• Design a memory strand with K = B + A bit regions.
                                                    Bits 1 ··· B represent which bags are
                                                    chosen, bits B + 1 ··· B + A which
• Initialize a (K, B) library set in a tube called T0.   object types are present.
• for i = 1 to B                                    Mark the final A positions of each
      Separate T0 into Ton and Toff based on bit i  complex to record which object types
      for j = 1 to |Ci|                             it contains.
          Set bit B + Ci[j] in Ton
      Combine Ton and Toff into T0
• for i = B + 1 to B + A                            Get rid of ones which do not have all
      Separate T0 into T0 and Tbad based on bit i   A types.
      Discard Tbad
• for i = 0 to B − 1                                Count how many bags were used. At
      for j = i down to 0                           the end of the outer loop, tube Ti
          Separate Tj into T(j+1)' and Tj based on bit i + 1   contains all complexes which used
          Combine Tj+1 and T(j+1)' into Tj+1        exactly i bags.
• Read T1;
  else if it was empty then    Read T2;
  else if it was empty then    Read T3;
                               ...
```

where above $|C_i|$ is the number of items in subset C_i and $C_i[j]$ is the j^{th} item in subset C_i. Note that the above algorithm takes $O(AB + B^2)$ steps, and $O(AB)$ bits to specify the problem.

We point out that, as we will envision a robotic system performing the experiments automatically, we allow arbitrary sequential algorithms for controlling the molecular operations. However, these operations must be performed "blind"; the only interface to molecular parallelism is via *initialize, combine, separate, set, clear,* and *read.* Thus the electronic algorithms are responsible for "experiment design," i.e., compiling higher-level problem specifications into concise sequences of molecular operations but they cannot get any feedback from the DNA during the course of the experiment.

As a final comment, we note that the stickers model is capable of simulating (in parallel) independent universal machines, one per memory complex, under the usual theoretical assumption of an unbounded number of sticker regions.[3] It should be noted that the stickers model is universal, in the sense discussed, even in the absence of the *clear* operation. Any algorithm that takes advantage of the *clear* operation can be rewritten to instead use *write-once* bits. This does, however, happen at a cost. Naively, an algorithm implemented without *clear* requires the addition of one *write-once* sticker region to the memory strand for each use of *clear* in the original algorithm. This means that, depending on the algorithm under consideration, memory strands may have to be much longer.

3. PHYSICAL IMPLEMENTATION OF THE MODEL

Each logical operation in our model has a corresponding interpretation (which we gave as we introduced the operations) in terms of what must happen to the DNA memory strands and associated stickers when that operation is carried out. In what follows we examine various physical procedures which are candidates for implementing these requirements for all the operations described above. We speak in terms of *tubes* instead of *sets*; recall that a *tube* consists of the collection of memory complexes that represents a *set* of bit strings.

Often, there are several possible implementations of a given operation; each has its own assumed strengths and weaknesses on which we speculate. However, which implementations, if any, turn out to be viable will ultimately have to be decided by laboratory experiments.

[3]This can be seen as the consequence of two observations. First, a memory complex in the stickers model can simulate a feedforward circuit, in the spirit of (Boneh *et al.*, 1996). Using the *clear* operation, a clocked feedback circuit can also be simulated. Second, allowing the circuit to grow with each clock cycle, we can simulate a universal machine. The electronic algorithm is responsible for designing the new gates to fit into the circuit; each new gate will require a new bit and hence a new sticker region in the memory strand. For concreteness, a feedforward circuit C_t can be automatically designed which computes the instantaneous description of a TM at time step t from the description at $t - 1$. Thus, the stickers model can simulate in parallel the execution of a TM on all 2^L length L inputs.

3.1. Combination

Combination of two tubes can be performed by rehydrating the tube contents (if not already in solution) and then combining the fluids together (by pouring or pumping for example) to form a new tube. It should be noted that even this seemingly straightforward operation is plagued by constraints: if DNA is not handled gently, the shear forces from pouring and mixing it will fragment it into ≈15 kilobase sections (Kornberg, 1992).

Also of concern for this operation and indeed for all others is the amount of DNA which remains stuck to the walls of tubes, pumps, pipette tips, etc. and thus is "lost" from the computation. Even if this "lost" DNA is a minute fraction of the total (which would be unimportant to molecular biologists) it is problematic for computation because we are working with relatively few copies of each relevant molecule.

3.2. Separation

The ultimate goal of the separation operation is to physically isolate those complexes in a tube that have a sticker annealed to some position from those that do not without disturbing any annealed stickers. The mechanism of DNA hybridization will be central to any proposal. In general, separation by hybridization is performed by bringing the solution containing the original set of memory complexes into contact with many identical single stranded *probes*. In our case, each bit position has a particular type of probe (with a unique nucleotide sequence) that is used when separation on that bit is performed. The probe sequence is designed such that probes hybridize only to the region of the memory strand corresponding to their bit and nowhere else. During separation, the original complexes with the key bit *off* will be captured on the probes while all those with the bit *on* will remain unbound in solution because the region is covered by a sticker. Next, the unbound ("on") complexes are physically isolated, for example by conjugating the probes to magnetic beads or affixing the probes to solid support and then washing. Lastly, the "off" memory complexes are recovered from the probes that bound them by elution (say by heating and washing). The result is two new tubes, one containing the memory complexes for each of the output sets of the operation.

Notice that if heating is used to achieve the final step of elution this must be done without also removing all of the stickers from the memory strands. This necessitates that the probes have a lower binding affinity for their corresponding regions than do the stickers. This might be achieved by making the probe sequences not exactly complementary to their regions on the memory strands (or merely shorter) to create a differential between the temperature of probe-strand and sticker-strand dissociation. An alternative is to use perfectly complementary sequences for both the probes and stickers but to make the stickers out of an alternate backbone material (such as PNA or DNG; Egholm *et al.*, 1993; Dempcy *et al.*, 1995), which would exhibit stronger and more specific binding to the DNA memory strand than DNA probes.[4] PNA and DNG offer the additional advantage that decreasing salt concentration causes PNA/DNA and DNG/DNA to bind more strongly, while the opposite is true for DNA/DNA binding. Thus the final elution step might be achieved by washing in a zero salt solution rather than by heating. There are other possibilities for creating differential affinity between the stickers and probes.[5]

3.3. Setting and clearing

To *set* a bit in every string of a set the most obvious choice is direct annealing. An excess amount of the sticker corresponding to the bit is added to the tube containing the set's memory complexes. One sticker should anneal to every complex that does not already have one, always in the position opposite the region corresponding to the bit being set. Subsequently the excess (unused) stickers are removed, perhaps by filtration or by *separating* out all the memory complexes. This latter proposal could be achieved by having a *universal region* on every memory strand (say, at the very beginning or end) that is never covered by a sticker and designing a probe for that region as described in the separation operation above. Such a universal region is a generally useful idea for recovering all memory complexes from a given solution which may contain other species.

---

[4]PNA "clamps" (Egholm *et al.*, 1995) have been shown to form $(PNA)_2$/DNA triplexes with remarkable affinity and specificity. These clamps could also be used as stickers.

[5]For example crosslinking techniques might be used to covalently bond the stickers to the memory strands so that they could not come off during elution, although this confounds the *clear* operation and does not keep with the reusable spirit of the model.

To *clear* a bit in every string of a set requires removing the stickers for only that bit from every complex in a tube. Simple heating will obviously not work since *all* stickers from *all* bit regions will come off. One possibility is to designate certain bit regions as *weak* regions. These regions have weak stickers which dissociate more easily from the memory strand than regular stickers. By heating to some intermediate temperature *all the weak stickers* can be made to dissociate at once, keeping all of the regular stickers in place.

In order to implement the *clear* operation in full generality, it may be possible to use the phenomenon of PNA strand invasion by triple helix formation (Nielsen *et al.*, 1991). It has been shown that under appropriate conditions, two single stranded oligos of all-pyrimidine PNA will "invade" an existing complementary DNA/DNA duplex to form a $(PNA)_2$/DNA triple helix, displacing the pyrimidine DNA strand. This process is most efficient with PNA "clamps" (Egholm *et al.*, 1995) which contain both the Watson-Crick and Hoogsteen PNA strands in a single molecule. We suggest that if, for example, 21-nucleotide DNA stickers are used, then a 14-base PNA clamp could be designed that forms a triple helix with the central 7 nucleotides of the DNA sticker. By mixing PNA clamps specific to a particular bit with a tube of memory complexes, and heating, the PNA clamps should form triple helices with the targeted sticker, destabilizing and thus "prying" it off at a temperature lower than the dissociation temperature for the unaffected stickers. The specificity and reliability of this operation are not yet known experimentally; indeed, the mechanism of triplex formation (Demidov *et al.*, 1995) may be incompatible with the requirement that non-targeted stickers remain in place. This is only one of many possible methods for implementing the *clear* operation, however it appears that in terms of physical implementation prospects, *clear* is the most problematic of our operations. Recall, however, that it can be eliminated without significantly sacrificing the computational power of the model.

3.4. Initialization and final output

To make a combinatorial library containing roughly one copy of every possible bit string of length L followed by $K - L$ zeros, it is first necessary to synthesize roughly 2^L identical copies of a properly designed memory strand with $K \geq L$ regions. Stickers must then be added "randomly" to these strands in positions $1 \cdots L$. One procedure that achieves this is outlined below. Note that the method requires only a single step.

The strands are split into two equal volumes. To one volume is added an excess of stickers for all bits $1 \cdots L$; this results in all bits $1 \cdots L$ being set on all strands. The unused stickers are then removed, for example by filtration or by separating on a universal region of the memory strand. The two volumes are then recombined and heated causing all stickers to dissociate. Finally the mixture is cooled again, causing the stickers to randomly anneal to the memory strands. Since each bit position has only one sticker for every two strands, the resulting memory complexes have any given bit set with probability one half (very nearly independently). Under this model, the odds that any particular bit string is *not* present in the final library is $(1 - 1/2^L)^{2^L}$ which for the L of interest is almost exactly $1/e$. In other words each string is created at least once with probability roughly 63%. This percentage can obviously be increased by synthesizing more than 2^L strands initially. Notice that this procedure is relatively robust to errors in stoichiometry: For example, if the original strands are split into volumes whose ratio is not 1 but 1.5 then (for say $L = 56$) a randomly chosen string is created with probability 37%, still not vanishingly small.[6]

To obtain an output string it is necessary to be able to *detect* the presence or absence of memory complexes in a solution. If any are present, we also need to be able to isolate at least one memory complex and then identify which stickers (if any) are annealed to it.

Detection of complexes might be accomplished by fluorescent labeling of each memory strand. Single molecule detection can then be performed by running the solution through a fine capillary tube. Such detection has already been achieved experimentally, see for example (Castro and Shera, 1995). This technique may also be effective for isolating a single complex if the time between detection events is large enough. In addition to the capillary tube method mentioned above, other proposals (e.g., based on PCR) for complex detection are possible.

The final step of identifying annealed stickers may be possible by direct imaging; since we know the order of bit regions, we could imagine just *looking* and reading off the answer string (perhaps using electron microscopy). Alternatively, once a complex is isolated its stickers may be eluted and poured over a detection hybridization grid (Meade, 1995) to determine which ones were present. While these possibilities are

[6]The expression for the probability of a random bit string being created is $1 - \sum_{k=0}^{L} \frac{\binom{L}{k}}{2^L}[1 - r^k(1+r)^{-L}]^{2^L}$, where r is the ratio of the volumes into which we split initially.

intriguing, more practical approaches based on PCR are more likely to work in the near term (Adleman *et al.*, 1996). However, we show below that detection alone is sufficient to obtain an output string. The approach is to use binary tree decoding:

Begin with the solution containing all putative answer complexes (of which there may be none). Detect complexes in it. If there are none, then no answer has been found. If there are some then separate them based on the first bit of the answer string.[7] Detect complexes in each of the resulting solutions and retain the one which is not empty. If neither is empty then there is more than one answer and either can be retained. Repeat this separation and detection for all the bits of the answer string.

3.5. Memory strand and sticker design

At several points in the above discussion, it was necessary to design the sequence of the memory strand or stickers to have certain properties. In this section, we summarize those requirements and explore possibilities for achieving them.

The most fundamental requirement of sequence design is to achieve sticker specificity. It is critical that the stickers only anneal to the memory strands when opposite their assigned region and not in any other position. Thus, the memory strand sequence must be designed so that any region's complementary sticker is only complementary to that one region and has much reduced affinity at all other alignments along the strand. As a first approximation to this, we will require a certain minimum number of base mismatches at all other alignments. Notice that this is a much stronger requirement than simply requiring each sticker to mismatch all bit regions but its own. It must mismatch every other M long window (possibly spanning two bit regions) on the strand. Mathematically, we wish to design a sequence of length N such that there exist K nonoverlapping subsequences of length M each (call them "regions") with the following property: For each region, its complement has at least D_1 mismatches with *every* other subsequence of length M in the entire sequence. The quantity D_1 is the minimum number of mismatches needed for a sticker M bases long not to anneal.

It is also important to eliminate secondary structure in the memory strand itself. We must prevent the memory strand from annealing to itself and creating a hairpin structure, as this makes regions inaccessible for proper use in the system. Fulfilling this requirement can be loosely modeled by the combinatorial problem of designing a N long sequence such that the complement of *every* subsequence of length M has at least D_2 mismatches with every other subsequence of length M. The quantity D_2 is the minimum number of mismatches to prevent the memory strand from self-annealing.

Finally, we must design separation probes such that they stick specifically to the appropriate region and they have sufficiently lower affinity there than the stickers. This ensures that there exists a wash temperature (and salinity) for which the probes will dissociate while the stickers will remain in place. Again, as a first approximation we require that the probes have at least D_3 mismatches within their region and at least $D_4 > D_3$ mismatches everywhere else.

These criteria may seem daunting. However, there are some ways to make this task potentially easier. Notice that in general we may leave portions the memory strand unused; that is we may not identify those portions with any regions so that the product of K and M does not always equal N (but certainly still $KM \leq N$). In other words, we leave "gaps" between the bit regions on the memory strand. In order to avoid the secondary structure problem, it has been suggested that the memory strand be composed of only pyrimidines and the stickers of only purines (Mir, 1999).[8] The applied mathematics literature on "comma free codes" and on "de Bruijn sequences" (when $D = 1$) contains detailed discussions of many of the important issues (for introductions, see Neveln, 1990; Fredricksen, 1982). Also, Smith (1996), Baum (1999), Frutos *et al.* (1997), Deaton *et al.* (1998), and (Hartemink and Gifford, 1997) have discussed sequence design in the context of DNA computation.

Finally, D_1 would be reduced if higher affinity PNA or DNG stickers were used; furthermore, D_3 would possibly be reduced to zero. Other variables other than or in addition to temperature could be manipulated, such as salt concentration and chemical solvent, in order to achieve the relative affinities required for each operation. It is worth speculating about the possibility of using naturally occurring sequences (e.g., plasmids)

[7] The answer string which we are interested in reading out may be a substring of the entire string encoded by the memory strand in which case separation only needs to be done for those bits.

[8] Long single-stranded polypyrimidine DNAs evidently hybridize without difficulty to short polypurine probes in the context of sequencing by hybridization, an endeavour plagued by DNA secondary structure.

for the memory strands because of the obvious ease of their mass production. However, it remains to be seen if natural sequences can be found which meet the above restrictions.

We emphasize that the criteria outlined above are for illustration only; a more sophisticated approach would have to take into consideration the sequence-dependent thermodynamic parameters for oligonucleotide hybridization. There are several data sets available for calculating ΔH and ΔS for DNA/DNA hybridization (Santalucia *et al.*, 1996; Breslauer *et al.*, 1986; Petruska and Goodman, 1995), and similar data could be obtained for PNA and/or DNG interactions. Allowances would also have to be made for potential bubble mismatches at incorrect sticker hybridization sites, and secondary structure due to triple helix formation must be prevented. The *clear* operation, if used, would introduce additional constraints. Although such sophisticated design approaches could suggest potentially useful memory strand, sticker, and probe sequences, correct operation will have to be tested experimentally.

Our conclusion is that although design of the memory strand and the stickers may be difficult, the design space is large; and once a strand with K regions is found, it can be used and reused in the stickers model for any problem requiring K or fewer bits of memory. Since the stickers model uses only a single type of memory strand, in contrast to the 2^K different molecules required in the representation of (Boneh *et al.*, 1996), the design process is simplified and the functionality of the strand can be tested experimentally once and for all.

3.6. *Experimental feasibility*

The stickers model as presented above presents challenging requirements for strand design and experimental implementation. Several objections might be raised to the effect that it is unreasonable to expect that these requirements can be met. We attempt to briefly address some of these issues here.

Objection: No matter what methods are proposed, DNA-based techniques will suffer from strands being misprocessed. What error rates would be required in order to still accomplish useful computation?

Response: For many search problems, including DES and NP-complete problems, probabilistic algorithms have practical value. Answers suggested by the molecular computer, so long as there aren't too many, can be verified electronically. To decrease the number of false-positive distractors, it may be necessary to refine the "answers" tube by repeating the steps of the computation (Adleman, 1996). If *no answer* is suggested by the molecular computer, then our understanding of the separation error probabilities will dictate our confidence in this "null result." To ensure that a complex carrying the solution to the problem has a 90% chance of ending up in the "answers" tube after a 1000-step computation, separation error probabilities of less than 0.01% are required. These and other related error-handling strategies are discussed in detail elsewhere (Karp *et al.*, 1995; Roweis and Winfree, 1999).

Objection: Purity and yield of 90% for purification of DNA are considered excellent in molecular biology. The conditions imposed for separation of memory complexes are much more challenging, since long strands may be used, stickers must not be knocked off, and both supernatant and eluant are required. Yet DNA computation requires much lower error rates, both for purity and yield.

Response: Isolation of particular target DNA in complicated cDNA libraries is a routine task in molecular biology. 10^5-fold enrichment of target DNA, with 80% recovery, has been reported using, for example, triplex affinity capture (Ito *et al.*, 1992). The use of PNA probes also shows some promise: 99% purification with 50% yield using PNA 15-mers has been reported (Orum *et al.*, 1995). However, current techniques do not meet our requirements for the separation operator. We do not believe that this is due to a fundamental limit. So long as yield is extremely high (i.e., memory complexes don't get "lost"), our calculations (Roweis and Winfree, 1999) suggest that a poor separation can be improved dramatically by automated processing. Furthermore, we have the opportunity to design our own sequences that can be effectively separated, for example, by ensuring that the memory strand has no secondary structure. We recognize that attaining high step yield may be a major challenge, however.

Objection: Even without trying to process them at all, stickers will be falling off their memory strands at some rate k_d. Once a sticker dissociates, it may then hybridize to and thus corrupt some other complex. During operations such as separate, when memory complexes must be melted from probes, k_d surely increases. By the time the computation is complete, the contents of the memory complexes may be completely scrambled.

Response: Suppose we would like to ensure that fewer than 0.01% of stickers fall off during the course of a 1,000-hour computation. This would require a k_d of less than $0.3 \times 10^{-9}/sec$. A generic DNA 20-mer can

be estimated to have the required k_d at 42°C in 1 M [Na^+] (Wetmur, 1991). PNA and DNG stickers would be expected to have an even lower dissociation rate, especially at low salt. High wash temperatures may be avoided by using DNA probes and PNA or DNG stickers, and washing in low salt. Additionally, we must be careful not to encourage other circumstances, such as rough physical handling, which might induce sticker dissociation.

Objection: If DNA is subjected to high temperatures for a significant portion of a 1,000-hour computation, it may be damaged by deamination, depurination, or strand breakage by hydrolysis, thus rendering it nonfunctional. (Such objections are discussed briefly in, for example, Smith, 1996.)

Response: Under physiological conditions of salinity, pH, and temperature, the depurination half-life of a base is 1,000,000 hours, and the hydrolysis half-life of a depurinated base is 400 hours (Friedberg, 1995). Thus, after 1,000 hours, approximately 0.1% of bases will be damaged, and 10% of 2,500-mer strands will remain unbroken. This last figure is very dependent on the length of the strands; only 0.1% of 5,000-mers would survive. While not good, this indicates that for "short" strands, errors due to damage can be compensated for by a mild increase in the volume of DNA used in a computation. Additionally, improved rates may be possible by carefully adjusting solvent salinity, pH, and composition—and again minimizing rough physical handling.

In summary, although there are many serious engineering challenges, we do not see any as being clearly insurmountable.

4. A STICKERS MACHINE PROPOSAL

This section describes the details of one possible machine that implements computation using the stickers model. The machine is a sort of "parallel robotic workstation for molecular computation" in which various robotic and fluid flow apparatuses are controlled by a central programmable electronic computer. It contains of a rack of many test tubes, a small amount of robotics, some fluid pumps and heaters/coolers, and some conventional microelectronics. For each of the operations in the model, we have made a specific choice of physical procedures to implement it. Thus, the machine represents one particular realization of many possible variations on the ideas discussed above. The proposal is meant to provoke thought about the engineering issues involved in eventually constructing a molecular computer and not as a serious or viable construction plan.

The workstation stores all DNA which represents information during the computation in so-called *data tubes*. Each data tube is a closed cylinder with a nipple connector in either end that allows fluid to flow in or out. Near one end on the inside is a permanent membrane which passes solvent but not stickers or memory strands. This membrane gives a polarity to the data tube: the connector on the end closest to the membrane is the "clean" side while the opposite connector is "dirty." No DNA is ever present on the clean side or in the clean connector. When a data tube is not in use, it is held clean side down with all of the DNA in the tube resting on the membrane.

The data tubes (which may be empty) hold either sets of memory complexes or supplies of unbound stickers. Specifically, each set of bit strings has associated with it a data tube that holds the memory strands and annealed stickers representing those strings. Also, each bit has associated with it a data tube that contains a supply of stickers corresponding to that bit.

Whenever a new set of complexes is created (e.g., from a separation operation), it is placed in a new data tube. Whenever a set of complexes is destroyed (e.g., from a combination operation), the data tube that used to contain it is discarded (or perhaps vigorously washed and sterilized for reuse).

In addition to data tubes, there also exist *operator tubes* of similar external construction but with different internal contents. A "blank" operator tube is merely an empty tube with nipple connectors on each end. A "sticker" operator tube is identical except for a permanent filter on its inside which passes stickers but not memory strands. A "separation" operator tube contains many identical copies of one bit's oligo probe. (There is a different separation operator tube for each bit.) It is designed so that the probes cannot escape from the tube but unbound memory complexes can. For example, the probes might be fastened to solid support (by biotinylating them and using a biotin binding matrix) or to large beads with filters that pass memory strands but not beads. For all of the operator tubes, both ends are considered "dirty." Figure 3 illustrates the data and operator tubes.

At any time during the operation of the machine, some tubes are in use and other are not. All tubes that are not in use are stored on a large rack or carousel. Any single operation takes place as follows: under control of

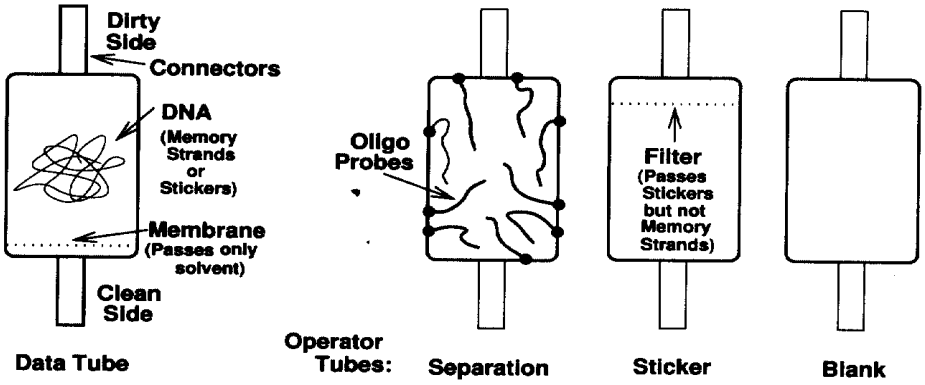

FIG. 3. Data and operator tubes in the stickers machine.

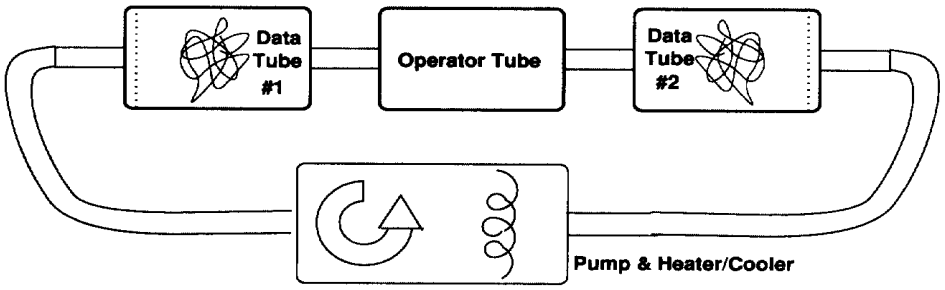

FIG. 4. Setup for a generic operation in the stickers machine.

the electronic computer two data tubes are selected and removed from the rack by a robot. One operator tube is also selected and removed. The dirty sides of the data tubes are connected to the operator tube, one data tube at each end of the operator. The clean sides of the data tubes are joined by a pump. Solution is cycled through all three tubes. The direction of flow may be towards the first data tube, or vice versa, or both intermingled. The temperature, salinity, direction, and duration of the flow is controlled by the electronic computer. Once the flow stops, one or more of the tubes is disconnected and replaced on the rack (or discarded). New tubes then come in from the rack until there are once again two data tubes and one operator tube and the next operation begins. Notice that in general clean connectors never touch dirty ones and only clean connectors contact the pumping system. This setup for a generic operation is shown in Figure 4.

We will now review how each of our conceptual operations can be performed as outlined generically above. The descriptions below are summarized graphically in Figure 5.

- To combine two sets of complexes simply select the two data tubes and a blank operator tube. Cycle cold solution towards (say) the first data tube. This catches all the memory complexes in the first data tube. The second data tube and the blank operator are discarded.
- To separate a set of complexes based on the value of some bit, select the data tube containing the complexes to be separated and also an empty data tube. Select the separation operator tube for the bit in question. Cycle cold solution in both directions for some time; this allows the probes to bind those complexes that have the bit in question off. Next cycle cold solution towards the empty data tube, forcing all the unbound memory complexes into it. Detach this (originally empty) tube and return it to the rack; it holds the complexes with the bit in question on. Replace it with another empty data tube. Cycle hot solution (or perhaps low salinity solution) towards this new data tube. This releases the memory complexes bound to the probes and forces them into the new data tube. Detach this tube and return it to the rack also; it contains complexes with the bit off. Discard the original data tube (now empty) and return the operator tube to the rack.
- To set a bit (add a sticker to a set of complexes), select the data tube containing the complexes and also the data tube containing the sticker supply for the sticker to be added. Using the sticker operator tube, cycle cold solution in both directions for some time. This washes the stickers over the memory complexes allowing them to anneal. Now cycle cold solution towards the sticker data tube. This returns the unused stickers and leaves all the memory complexes caught on the filter in the operator tube. Disconnect the sticker data

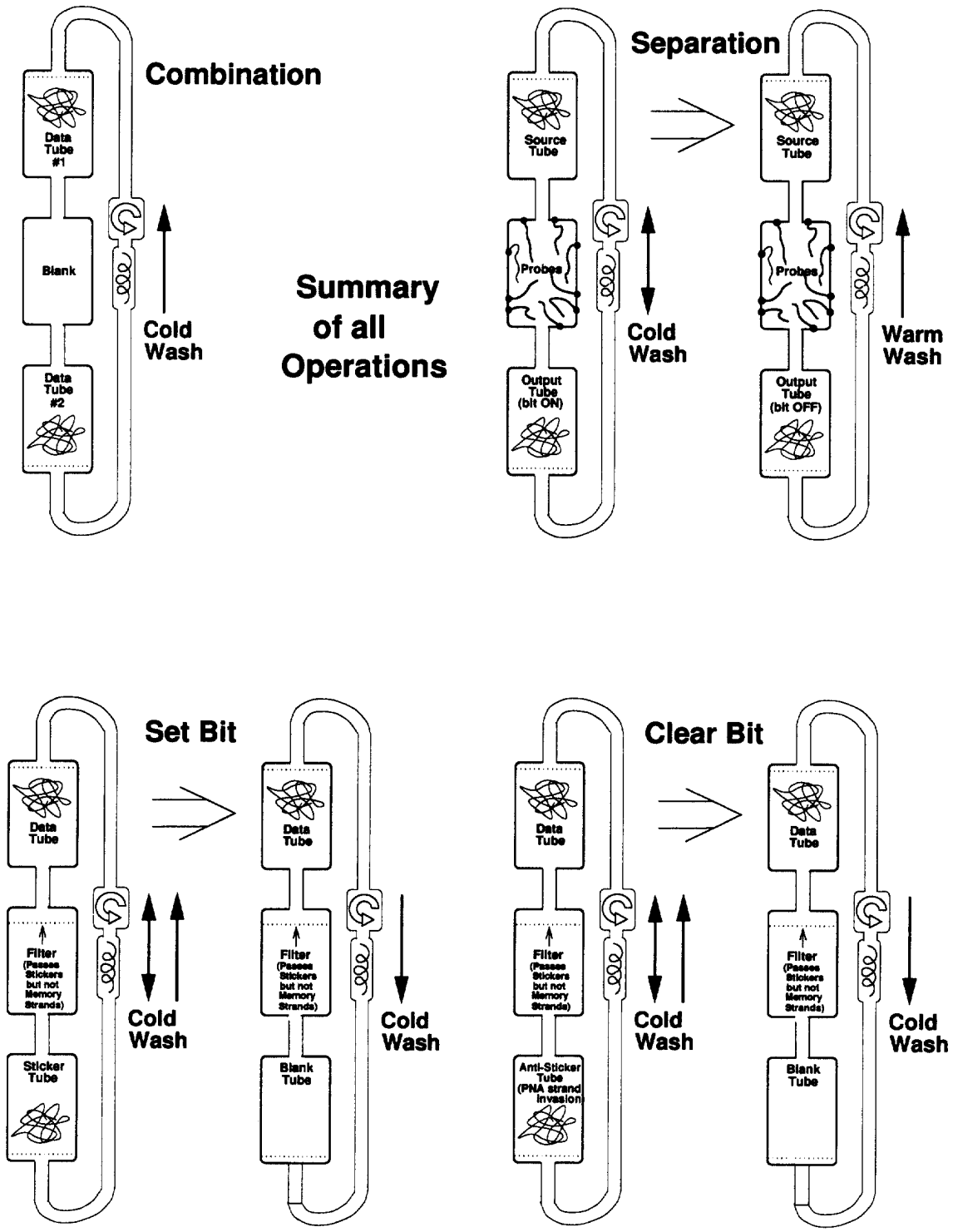

FIG. 5. Graphical synopsis of all operations in the stickers machine.

tube and return it to the rack. Replace it with an empty data tube. Cycle cold solution towards the memory complex data tube. This expels the memory complexes from the operator tube and returns them to their data tube. Return the memory complex data tube to the rack, and discard the operator tube and empty data tube.

Additional parallelism can be added in many places. For example, setting or clearing bits might be applied to many data tubes at once by stacking all of them after the operator tube. Also, many copies of the robotics

might be included to allow several operations to be performed simultaneously (this would also require multiple copies of, for example, the separator operator and sticker operator tubes).

As we have described it, the stickers machine requires relatively rudimentary robotics and electronics. Simple fluid pumps and heaters/coolers are also necessary. It can be stocked with a generic supply of empty data tubes, blank operator tubes, sticker operator tubes, and salt solutions of various concentrations. It contains data tubes containing both the original sets of memory strands and the sticker supplies for each bit. It also needs to be loaded with the separation operator tubes for each bit. An important feature is that these tubes are reusable from problem to problem, so long as the number of bits required does not exceed the number of regions on the designed memory strand. For a problem of reasonable size, on the order of a few thousand tubes might be required (for example, DES as described in Adleman *et al.*, 1999). With each data tube being a few millimeters in size and operator tubes perhaps a hundred times this size, it is not inconceivable that such a machine might fit on a desktop or lab bench. This example directly addresses the concern that any useful or hard computation will require an enormous volume of DNA by demonstrating both a specific problem and a specific machine proposal for which this seems far from true.

5. CONCLUSION

In this paper, we have tried to visualize a practical molecular computer. A number of previous concerns (Smith, 1996; Hartmanis, 1995; Linial *et al.*, 1995) have been addressed. First, it is now clear, from our own work and that of others, that general-purpose algorithms can be implemented by DNA-based computers, potentially solving a wide class of search problems. Second, we now understand that there are challenging problems, such as breaking DES, for which only modest volumes of DNA (e.g., 2 grams) should suffice. Third, we demonstrated that the formation and breaking of covalent bonds is not intrinsic to DNA-based computation. All the materials in the stickers model are potentially reusable from one computation to the next. Fourth, we have shown that a single essential biotechnology, sequence-specific separation, suffices for constructing a general-purpose molecular computer. We also know from other work (Karp *et al.*, 1995; Roweis and Winfree, 1999) that separation errors can theoretically be reduced to tolerable levels by invoking a trade-off between time, space, and error rates at the level of algorithm design.

That several major roadblocks have been overcome at a theoretical level suggests that real applications of molecular computation may be feasible in the future. Nonetheless, we emphasize that substantial engineering challenges remain at almost all stages and that the ultimate success or failure of DNA computing will certainly depend on whether these challenges can be met in laboratory investigations.

ACKNOWLEDGMENTS

We would like to express their appreciation to Professor John Baldeschwieler for his contributions to this paper through early discussions of this work. S.R. and E.W. are also grateful to their advisor, Professor John Hopfield, for his perpetual wisdom and long-term advice. Thanks to Kazuyoshi Harada for some corrections. S.R. is supported in part by the Center for Neuromorphic Systems Engineering as a part of the National Science Foundation Engineering Research Center Program under grant EEC-9402726 and by the Natural Sciences and Engineering Research Council of Canada. E.W. is supported in part by National Institute for Mental Health (NIMH) Training Grant no. 5 T32 MH 19138-06; also by General Motors' Technology Research Partnerships program. L.M.A., N.V.C., and P.W.K.R. are supported in part by grants from the National Science Foundation (CCR-9403662) and Sloan Foundation.

REFERENCES

Adleman, L.M. 1994. Molecular computation of solutions to combinatorial problems. *Science* 266, 1021–1024.

Adleman, L.M. 1996. On constructing a molecular computer. *In* Lipton, R.J., and Baum, E.B., eds., *DNA-Based Computers: DIMACS Workshop, April 4, 1995.* Volume 27 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 1–21.

Adleman, L.M., Rothemund, P.W.K., Roweis, S., and Winfree, E. 1999. On applying molecular computation to the data encryption standard. *In* Landweber, L.F., and Baum, E.B., eds., *DNA-Based Computers II: DIMACS Workshop, June 10–12, 1996.* Volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 151–162.

Amos, M., Gibbons, A., and Hodgson, D. 1999. Error-resistant implementation of DNA computations. *In* Landweber, L.F., and Baum, E.B., eds., *DNA-Based Computers II: DIMACS Workshop, June 10–12, 1996.* Volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 151–162.

Baum, E.B. 1999. DNA sequences useful for computation. *In* Landweber, L.F., and Baum, E.B., eds., *DNA-Based Computers II: DIMACS Workshop, June 10–12, 1996.* Volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 235–242.

Beaver, D. 1995. Molecular Computing. *Technical Report CSE-95-001*, Penn State University.

Boneh, D., Dunworth, C., Lipton, R.J., and Sgall, J. 1996. On the computational power of DNA. *Discrete Appl. Math.* 71(1–3), 79–94.

Breslauer, K.J., Frank, R., Blöcker, H., and Marky, L.A. 1986. Predicting DNA duplex stability from the base sequence. *Proc. Natl. Acad. Sci. U.S.A.* 83, 3746–3750.

Castro, A., and Shera, E.B. 1995. Single-molecule detection: applications to ultrasensitive biochemical analysis. *Appl. Optics* 34(18), 3218–3222.

Deaton, R., Garzon, M., Murphy, R.C., Rose, J.A., Franceschetti, D.R., and Stevens, Jr., S.E. 1998. Reliability and efficiency of a dna-based computation. *Phys. Rev. Lett.* 80(2), 417–420.

Demidov, V.V., Yavnilovich, M.V., Belotserkovskii, B.P., Frank-Kamenetskii, M.D., and Nielsen, P.E. 1995. Kinetics and mechanism of polyamide ("peptide") nucleic acid binding to duplex DNA. *Proc. Natl. Acad. Sci. U.S.A.* 92, 2637–2641.

Dempcy, R.O., Browne, K.A., and Bruice, T.C. 1995. Synthesis of a thymidyl pentamer of deoxyribonucleic guanidine and binding studies with DNA homopolynucleotides. *Proc. Natl. Acad. Sci. U.S.A.* 92, 6097–6101.

Egholm, M., Buchardt, O., Christensen, L., *et al.* 1993. PNA hybridizes to complementary oligonucleotides obeying the Watson-Crick hydrogen bonding rules. *Nature* 365, 566–568.

Egholm, M., Christensen, L., Dueholm, K.L., Buchardt, O., Coull, J., and Nielsen, P.E. 1995. Efficient pH-independent sequence-specific DNA binding by pseudoisocytosine-containing bis-PNA. *Nucleic Acids Res.* 23(2), 217–222.

Fredricksen, H. 1982. A survey of full-length nonlinear shift register cycle algorithms. *SIAM Rev.* 23(2), 195–221.

Friedberg, E.C., Walker, G.C., and Siede, W. 1995. *DNA Repair and Mutagenesis*, ASM Press, Washington, DC, 13–14.

Frutos, A.G., Liu, Q., Thiel, A.J., *et al.* 1997. Demonstration of a word design strategy for dna computing on surfaces. *Nucleic Acids Res.* 25(23), 4748–4757.

Garey, M.R., and Johnson, D.S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, 222.

Hartemink, A.J., and Gifford, D.K. 1997. Thermodynamic simulation of deoxynucleotide hybridization for DNA computation. Presented at the 3rd DIMACS Meeting on DNA-Based Computers, University of Pennsylvania, June 23–25, 1997, 15–34. To appear.

Hartmanis, J. 1995. On the weight of computations. *Bull. Eur. Assoc. Theor. Comput. Sci.* 55, 136–138.

Ito, T., Smith, C.L., and Cantor, C.R. 1992. Sequence-specific DNA purification by triplex affinity capture. *Proc. Natl. Acad. Sci. U.S.A.* 89, 495–498.

Karp, R., Kenyon, C., and Waarts, O. 1995. Error-resilient DNA computation. Research report 95-20, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon.

Kornberg, A. 1992. *DNA Replication*, 2nd edition, W. H. Freeman & Co., New York, 21.

Linial, M., Linial, N., Lo, Y.-M.D., Yiu, K.F.C., Wong, S.L., and Bunow, B. 1995. Letters. *Science* 268, 481–484.

Lipton, R.J. 1995. DNA solution of hard computational problems. *Science* 268, 542–545.

Meade, T.J. 1995. *Sci. Am.* 33–34.

Mir, K.U. 1999. A restricted genetic alphabet for DNA computing. *In* Landweber, L.F., and Baum, E.B., eds., *DNA-Based Computers II: DIMACS Workshop, June 10–12, 1996.* Volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 243–246.

Neveln, B. 1990. Comma-free and synchronizable codes. *J. Theor. Biol.* 144, 209–212.

Nielsen, P.E., Egholm, M., Berg, R.H., and Buchardt, O. 1991. Sequence-selective recognition of DNA by strand displacement with a thymine-substituted polyamide. *Science* 254; 1497–1500.

Orum, H., Nielsen, P.E., Jorgensen, M., Larsson, C., Stanley, C., and Koch, T. 1995. Sequence-specific purification of nucleic acids by PNA-controlled hybrid selection. *BioTechniques* 19, 472–480.

Petruska, J., and Goodman, M.F. 1995. Enthalpy-entropy compensation in DNA melting thermodynamics. *J. Biol. Chem.* 270(2), 746–750.

Rothemund, P.W.K. 1996. A DNA and restriction enzyme implementation of Turing machines. *In* Lipton, R.J., and Baum, E.B., eds., *DNA-Based Computers: DIMACS Workshop, April 4, 1995.* Volume 27 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 75–119.

Roweis, S., and Winfree, E. 1999. Reducing error rates: A refinery model. *In* Landweber, L.F., and Baum, E.B., eds., *DNA-Based Computers II: DIMACS Workshop, June 10–12, 1996.* Volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 15–27.

Santalucia, J., Allawi, H.T., and Seneviratne, A. 1996. Improved nearest-neighbor parameters for predicting DNA duplex stability. *Biochemistry* 35(11), 3555–3562.

Smith, W.D. 1996. DNA computers in vitro and vivo. *In* Lipton, R.J., and Baum, E.B., eds., *DNA-Based Computers: DIMACS Workshop, April 4, 1995.* Volume 27 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, RI, 121–185.

Wetmur, J.G. 1991. DNA probes: applications of the principles of nucleic acid hybridization. *Crit. Rev. Biochem. Mol. Biol.* 36, 227–259.

Address reprint requests to:
Leonard M. Adleman
Department of Biological Sciences
University of Southern California
Los Angeles, CA 90089

adleman@pollux.usc.edu

Received for publication November 9, 1997; accepted as revised June 6, 1998.