the nature of their silicate inclusions is quite diverse, ranging from "primitive" objects (Techado) to highly differentiated materials (Colomera). A previous model for the origin of IIE iron meteorites that argues for the mixing of silicates in a cooling metal pudding (*26*) is unable to explain the presence of chondritic (unmelted) objects as inclusions in large masses of (molten) metal. It has also been suggested that the silicates became incorporated into segregated, low-temperature metallic melts that were separated by shock-induced shear transport (*27*). This model is supported by the evidence for a "nonmagmatic" origin of IIE metal (*27*). However, no shock effects (such as metal veins or silicates with undulating extinction) have been detected in the silicate inclusions. Although we cannot exclude the possibility that such effects were annealed during the subsolidus history of the breccias, the absence of shock features is in conflict with the suggestion that metal and silicates in IIE iron meteorites were mixed by impact melting. Another view invokes impact mixing of silicates with a metal core from a different parent body (*28*). This model offers a plausible explanation for the unshocked nature of the silicates but requires a magmatic origin of the metal.

It seems necessary to call on the existence of a partially melted H-chondrite asteroid to account for the variety of silicate inclusions observed in different specimens. Some areas of this body may have preserved their primitive compositions (for example, the inclusion in Techado), and others became highly enriched in incompatible elements as a result of the fractionation of silicate magmas of minimum melt composition (represented by trydimite and potassium feldspar crystals observed in Colomera and other IIE iron meteorites). It is possible that, as a consequence of partial melting, sulfur-rich metallic liquids locally segregated and became intermingled with silicates. It is not likely, however, that such silicate objects could preserve their original metal and sulfide contents if immersed in such a metallic magma. Watson contains a good example of a silicate body that was partially melted, losing essentially all its metal and troilite, but retained a relatively undifferentiated composition (*8*). Therefore, mixing of the metallic liquid and partially melted silicates must have been followed by very rapid cooling of the assemblage in order to preserve the unmelted chondritic inclusion. It appears from the radiogenic ages that the "old" subgroup IIE area of the parent body was not substantially reheated by subsequent impacts. Therefore, the inferred cooling rates (Fig. 2) imply that the breccias were buried at a considerable depth in a megaregolith. Given that breccia formation most probably took place on the surface, burial at such depths may have occurred through collisional mixing and accretion of the parental asteroids.

REFERENCES AND NOTES

1. Here, the terms "magmatic" and "nonmagmatic" are not synonymous with melted and unmelted, respectively, as commonly used in the geological nomenclature. They actually refer to different crystallization processes. However, for historical reasons, we use Wasson's (*2*) terminology in this paper.
2. J. T. Wasson, *Meteorites: Their Record of Early Solar-System History* (Freeman, New York, 1985), p. 81.
3. R. N. Clayton, T. K. Mayeda, E. Olsen, M. Prinz, *Earth Planet. Sci. Lett.* **65**, 229 (1983).
4. R. N. Clayton, T. K. Mayeda, J. N. Goswami, E. J. Olsen, *Geochim. Cosmochim. Acta* **55**, 2317 (1991).
5. E. Olsen and E. Jarosewich, *Science* **174**, 583 (1971).
6. C. B. Gomes and K. Keil, *Brazilian Stone Meteorites* (Univ. of New Mexico Press, Albuquerque, NM, 1980), p. 31.
7. R. W. Bild and J. T. Wasson, *Science* **197**, 58 (1977).
8. E. Olsen *et al.*, *Meteoritics* **29**, 200 (1994).
9. A. L. Graham, *ibid.* **19**, 55 (1984).
10. D. J. Malvin, D. Wang, J. T. Wasson, *Geochim. Cosmochim. Acta* **48**, 785 (1984).
11. V. F. Buchwald and R. S. Clarke Jr., *Am. Mineral.* **74**, 656 (1989).
12. H. Craig, in *Isotopic and Cosmic Chemistry*, H. Craig, S. L. Miller, G. J. Wasserburg, Eds. (North-Holland, Amsterdam, 1964), pp. 401–451.
13. J. Willis and J. I. Goldstein, *Proc. Lunar Planet. Sci. Conf. B* **12**, 1135 (1981).
14. K. Marti and Th. Graf, *Annu. Rev. Earth Planet. Sci.* **20**, 221 (1992), and references therein.
15. L. Schultz and H. Kruse, *Meteoritics* **24**, 155 (1989).
16. J. Masarik and R. C. Reedy, *Geochim. Cosmochim. Acta* **58**, 5307 (1994).
17. B. Lavielle and K. Marti, in preparation.
18. F. Begemann, H. W. Weber, E. Vilcsek, H. Hintenberger, *Geochim. Cosmochim. Acta* **40**, 353 (1976).
19. Th. Graf and K. Marti, in preparation.
20. In the silicate samples, a correction of the $^{22}Ne/^{21}Ne$ ratio for matrix effects is obtained by adjusting this shielding indicator to yield a value for the ^{21}Ne exposure age of 60 Ma. The resulting value is $^{22}Ne/^{21}Ne = 1.14$ for an H-chondritic matrix and may be compared with a predicted matrix-corrected value of >1.14 (*16*). On the basis of model calculations (*21*), this $^{22}Ne/^{21}Ne$ ratio restricts the shielding depth of the silicate samples to about 10 cm. Furthermore, iron meteorites with $^{4}He/^{21}Ne < 250$ were inferred to have preatmospheric radii of <50 cm (*22*).
21. Th. Graf, H. Baur, P. Signer, *Geochim. Cosmochim. Acta* **54**, 2521 (1990).
22. H. Voshage, *Earth Planet. Sci. Lett.* **71**, 181 (1984).
23. S. Niemeyer, *Geochim. Cosmochim. Acta* **44**, 33 (1980).
24. H. G. Sanz, D. S. Burnett, G. J. Wasserburg, *ibid.* **34**, 1227 (1970).
25. D. S. Burnett and G. J. Wasserburg, *Earth Planet. Sci. Lett.* **2**, 137 (1967).
26. G. J. Wasserburg, H. G. Sanz, A. E. Bence, *Science* **161**, 684 (1968).
27. J. T. Wasson and J. Wang, *Geochim. Cosmochim. Acta* **50**, 725 (1986).
28. M. Prinz, C. Nehru, J. Delaney, M. Weisberg, E. Olsen, *Lunar Planet. Sci.* **XIV**, 618 (1983).
29. Techado oxygen isotopic composition: $\delta^{18}O = 3.69$ per mil and $\delta^{17}O = 2.43$ per mil (R. N. Clayton, unpublished data).

30. We thank A. J. Brearley (Institute of Meteoritics, University of New Mexico) for providing the Techado specimen, K. V. Ponganis for assistance with noble gas measurements, R. N. Clayton for oxygen isotope analyses, J. Weinstein for photographic work, B. Strack for technical assistance with the scanning electron microscope, and T. J. McCoy for motivation and helpful comments. J. T. Wasson and an anonymous reviewer provided constructive criticism on the original manuscript. This work was partially funded by National Aeronautics and Space Administration grant NAGW 3428.

18 October 1994; accepted 27 February 1995

DNA Solution of Hard Computational Problems

Richard J. Lipton

DNA experiments are proposed to solve the famous "SAT" problem of computer science. This is a special case of a more general method that can solve NP-complete problems. The advantage of these results is the huge parallelism inherent in DNA-based computing. It has the potential to yield vast speedups over conventional electronic-based computers for such search problems.

In a recent breakthrough, Adleman (*1*) showed how to use biological experiments to solve instances of the Hamiltonian path problem (HPP): Given a set of "cities" and directed paths between them, find a directed tour that starts at a given city, ends at a given city, and visits every other city exactly once. This problem is known to be NP-complete (*2*); that is, all NP problems can be efficiently reduced to it. A computational problem is in NP provided it can be formulated as a "search" problem. Further, a problem is NP-complete provided that if it has an efficient solution, then so do all problems in NP. One of the major achievements of computer science in the last two decades is the understanding that many important computational search problems are not only in NP but are NP-complete. Another major achievement is the growing evidence that no general efficient solution exists for any NP-complete problem.

Thus, Adleman's result that HPP can be solved by a DNA-based biological experiment is exciting. However, it does not mean that all instances of NP problems can be solved in a feasible way. Adleman solved the HPP with brute force: He designed a biological system that "tries" all possible

Princeton University, Princeton, NJ 08540, USA. E-mail: rjl@princeton.edu

tours of the given cities. The speed of any computer, biological or not, is determined by two factors: (i) how many parallel processes it has and (ii) how many steps each can perform per unit time. For biological systems, the first of these factors can be very large: As little as 3 g of water contains approximately 10^{22} molecules. Thus, biological computations could potentially have vastly more parallelism than conventional ones.

The second of these factors is very much in the favor of conventional electronic computers: A state-of-the-art supercomputer can easily do 100 million instructions per second; on the other hand, a biological machine seems to be limited to just a small fraction of a biological experiment per second. However, the biological machine's advantage in parallelism is so huge that the difference in the execution time for one instruction is not a problem.

However, even this advantage in parallelism does not allow every instance of an NP problem to be solved feasibly: Even with 10^{23} parallel computers, one cannot try all tours for a problem with 100 cities. The brute force algorithm is simply too inefficient. Biological computers can solve any HPP of, say, 70 or less edges. However, a practical issue is that there does not seem to be a great need to solve such HPPs. It is possible to routinely solve much larger HPPs on conventional machines (although a conventional machine will fail on some graphs of 100 nodes).

One might be tempted to conclude that biological computations are only a curious footnote to the history of computing. This is incorrect; it is possible to use biological computations to speed up many important computations (*3*). In particular, the method of Adleman (*1*) can be extended in a way that allows biological computers to potentially radically change the way that we do all computations, not just HPPs. I will show how to solve another famous NP-complete problem, the so-called "satisfaction" problem (SAT). In (*3*), I showed how to solve essentially any problem from NP directly. The goal here is to present the full details of the results first sketched in (*3*).

SAT is a simple search problem that was one of the first NP-complete problems. Consider the formula

$$F = (x \vee y) \wedge (\bar{x} \vee \bar{y}) \qquad (1)$$

The variables x and y are Boolean: They are allowed to range only over the two values 0 and 1. Usually, one thinks of 0 as "false" and 1 as "true". Then, $\vee$ is the logical OR operation ($x \vee y = 0$ only if $x = y = 0$), $\wedge$ is the logical AND operation ($x \wedge y = 1$ only if $x = y = 1$), and $\bar{x}$ denotes the "negation" of x ($\bar{x}$ is 0 if $x = 1$ and 1 if $x = 0$). The SAT problem is to find Boolean values for x and y that make the formula F true. In this example, $x = 0$ and $y = 1$ works, as does $x = 1$ and $y = 0$, whereas $x = y = 0$ does not, nor does $x = y = 1$.

The formula F consists of two clauses: The first is $x \vee y$, and the second is $\bar{x} \vee \bar{y}$. A clause is a formula that is of the form $v_1 \vee \ldots \vee v_k$ where each v_i is a variable or its negation. In general, a SAT problem consists of a Boolean formula of the form $C_1 \wedge \ldots \wedge C_m$, where each C_i is a clause. The question is, then, to find values for the variables so that the whole formula has the value 1. This is the same as finding values for the variables that make each clause have the value 1. The reason for calling this problem the satisfaction problem is that making all of the clauses true is often called "satisfying" the clauses. The current best method essentially tries all 2^n choices for the n variables.

Our model of how DNA behaves is simple and idealized. It ignores many complex known effects but is an excellent first-order approximation (*4*). Strands of DNA are just sequences $\alpha_1, \ldots, \alpha_k$ over the alphabet {A, C, G, T}. Double strands of DNA consist of two DNA sequences, $\alpha_1, \ldots, \alpha_k$ and $\beta_1, \ldots, \beta_k$, that satisfy the Watson-Crick complementary condition: For each $i = 1, \ldots, k$, α_i and β_i must be complements, that is, A $\leftrightarrow$ T or C $\leftrightarrow$ G. Complementary sequences anneal in an antiparallel fashion, where 5′ and 3′ refer to the chemically distinct ends of the DNA strands

$$\begin{array}{ccccccccccc} 5' & - & \alpha_1 & - & \alpha_2 & - & \alpha_3 & \ldots & - & 3' \\ & - & \updownarrow & & \updownarrow & & \updownarrow & & & \\ 3' & - & \beta & - & \beta_2 & - & \beta_3 & \ldots & - & 5' \end{array} \qquad (2)$$

There are a number of simple operations that can be performed on test tubes that contain DNA strands. (i) First, it is possible to synthesize large numbers of copies of any short single strand (here short is at least 20 nucleotides, which is all that I will require). (ii) Second, it is possible to create a double strand of DNA from complementary single strands by allowing them to anneal. (iii) Third, given a test tube of DNA, one can extract those sequences that contain some consecutive pattern of length l. Assuming that the pattern is $\delta_1, \ldots, \delta_l$, where each δ_i is in {A, C, G, T}, a DNA strand $\alpha_1, \ldots, \alpha_k$ will be removed only if, for some i,

$$\delta_1 = \alpha_i, \delta_2 = \alpha_{i+1}, \ldots, \delta_k = \alpha_{i+k-1} \qquad (3)$$

I call this operation "extract." The reason I call this extract and not "separate," as others have suggested, is that in practice the operation only extracts some of the required strands (a typical value might be 90%). Because the operation is not "complete," the term extract may be more suggestive.

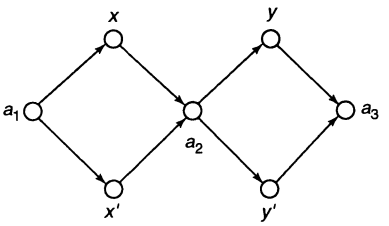

Fig. 1. The graph G_n, which encodes two-bit numbers.

(iv) Fourth, I assume that given a test tube, one can perform a "detect" operation. This operation simply determines whether or not there are any DNA strands at all in the test tube. It can be done by polymerase chain reaction (PCR) (*4*). (v) Finally, the last operation is called "amplify." This operation replicates all of the DNA strands in the test tube. [This operation was used in (*3*) but is not needed for the case of SAT.]

In our computations, I will always start with one fixed test tube; it is the same for all computations. This is an advantage over the method in (*1*). The set of DNA in this test tube corresponds to the following simple graph G_n. The test tube is formed in the same way that Adleman (*1*) formed the test tube of all paths to find the Hamiltonian path. The graph G_n has nodes $a_1, x_1, x'_1, a_2, x_2, x'_2, \ldots, a_{n+1}$ with edges from a_k to both x_k and x'_k and from both x_k and x'_k to a_{k+1} (Fig. 1). The paths of length $n + 1$ that start at a_1 and end at a_{n+1} are assumed to be in the initial test tube. This graph is constructed so that all paths that start at a_1 and end at a_{n+1} encode an n-bit binary number. At each stage, a path has exactly two choices: If it takes the vertex with an unprimed label, it will encode a 1; if it takes the vertex with a primed label, it will encode a 0. Therefore, the path $a_1x'a_2ya_3$ encodes the binary number 01 (Fig. 1).

Following (*1*), this graph is encoded into a test tube of DNA as follows. Each vertex of the graph is assigned a random pattern of length l from {A, C, G, T}. The length of $l = 20$, used in (*1*), should suffice here. This "name" of the vertex has two parts: The first half is denoted by p_i, and the second half by q_i. Thus, p_iq_i is the name associated with the ith vertex. Then, a test tube is filled with the following kinds of DNA strands:

1) For each vertex, put many copies of a $5' \rightarrow 3'$ DNA sequence of the form p_iq_i into the test tube.

2) For each edge from $i \rightarrow j$, place many copies of a $3' \rightarrow 5'$ DNA sequence of the form $\hat{q}_i\hat{p}_j$ ($\hat{x}$ denotes the sequence that is the Watson-Crick complement to x).

3) Add a $3' \rightarrow 5'$ sequence of length $l/2$ complementary to the first half of the initial vertex to the test tube. Similarly, add a 3′

$\rightarrow 5'$ sequence complementary to the last half of the final vertex to the test tube (that is, add $\hat{p}_1$ and $\hat{q}_n$).

The key is that every legal path in G_n corresponds to a correctly matched sequence of vertices and edges. Consider any path in the graph; it naturally consists of a sequence that alternates "vertex, edge, vertex, edge, ..." Suppose that $v \rightarrow u$ is an edge. Then, a path that passes through v and then u fits together like "bricks":

$$\begin{array}{lll} & v & u \\ (5' \rightarrow 3') & \overline{\qquad} & \overline{\qquad} \\ (3' \rightarrow 5') & \overline{\qquad\qquad} & \\ & v \rightarrow u & \end{array} \tag{4}$$

The top $5' \rightarrow 3'$ part consists of a series of "vertices." The bottom $3' \rightarrow 5'$ part consists of a series of "edges." The vertex v is coded as $p_v q_v$, and the edge is $\hat{q}_v \hat{p}_u$. The end of the vertex and the beginning of the edge can anneal because they are Watson-Crick complements. In the same way, the end of the edge and the beginning of the next vertex can also anneal. Moreover, because the sequences are chosen randomly, if l is large enough, then there is a high probability that no inadvertent paths will form. Thus, after annealing, there will be DNA encoding all of the paths through the graph; that is, it will encode all n-bit sequences.

This graph has one more important property. All of the paths are "similar": Each is different only in whether it goes "left" or "right" at a particular stage. Thus, there is no reason to believe that some paths will be more likely to appear than others. This is an important practical advantage: If only 99% of the paths are formed, then our method will have a 99% chance of success.

Operations are performed only on the DNA sequences from the graph G_n. I use $E(t, i, a)$ to denote all of the sequences in test tube t for which the ith bit is equal to a, for a in {0, 1}. This is done by performing one extract operation that checks for the sequence that corresponds to the name of x_i if $a = 1$ and to the name of x'_i if $a = 0$. The lengths of these names are long enough that it is unlikely that this sequence will occur by accident somewhere else in the piece of DNA. In some of the constructions, I use the remainder, that is, the strands that do not match the given pattern. Note that in all of the following, the strands of DNA can be assumed to be single strands.

Before proving our general result, let's try the example $F = (x \vee y) \wedge (\bar{x} \vee \bar{y})$ (Eq. 1). I construct a series of test tubes. The first one, t_0, is just the test tube of all two-bit sequences. Then, operate as follows:

1) Let t_1 be the test tube that corresponds to $E(t_0, 1, 1)$. Let the remainder be t'_1 and let t_2 be $E(t'_1, 2, 1)$. Pour t_1 and t_2 together to form t_3.

2) Let t_4 be the test tube that corresponds to $E(t_3, 1, 0)$. Let the remainder be t'_4. Let t_5 be $E(t'_4, 2, 0)$. Again pour t_4 and t_5 together to form t_6.

3) Check to see if there is any DNA in the last test tube, t_6. The satisfying assignments are exactly those in this final test tube.

To understand how this works, consider Table 1. Tube t_3 consists of all those sequences that satisfy the first clause: 01, 10, 11. In the same way, t_6 consists of all those from t_3 that satisfy the second clause: 01, 10. The latter are exactly the correct answers to the given SAT problem.

Let's now turn to the general case. Any SAT problem on n variables and m clauses can be solved with at most order m extract steps and one detect step. By "order m" I mean that the number of steps is linear in m. This assumes, as is usual, that each clause consists of a fixed number of variables or their negations. Let $C_1, \ldots, C_m$ be the clauses. A series of test tubes $t_0, \ldots, t_m$ are constructed so that t_k is the set of n-bit numbers x so that $C_1(x) = C_2(x) = \ldots = C_k(x) = 1$, where $C_i(x)$ is the value of the clause C_i on the setting of the variables to x. For t_0, use the set t_{all} of all possible n-bit numbers. Assuming t_k has been constructed, we construct t_{k+1}. Let C_{k+1} be the clause

$$v_1 \vee \ldots \vee v_l \tag{5}$$

where each v_i is a literal or a complement of a literal. For each literal v_i, operate as follows: If v_i is equal to x_j, then form $E(t_k, j, 1)$; if it is equal to $\bar{x}_j$, then form $E(t_k, j, 0)$. As in the example, the remainder of each extraction is used for the next step. Pour all of these together to form t_{k+1}. Then, do one detect operation on t_m to decide whether or not the clauses are satisfiable.

Table 1. Values encoded by the DNA in the test tubes during the biological solution of Eq. 1.

Test tube	Values present
t_0	00, 01, 10, 11
t_1	10, 11
t'_1	00, 01
t_2	01
t_3	01, 10, 11
t_4	01
t'_4	10, 11
t_5	10
t_6	01, 10

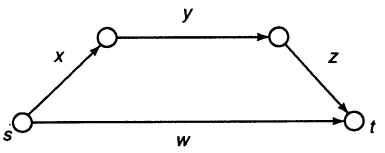

Fig. 2. A simple contact network. This network is satisfied only if $w = 1$ or $x = y = z = 1$.

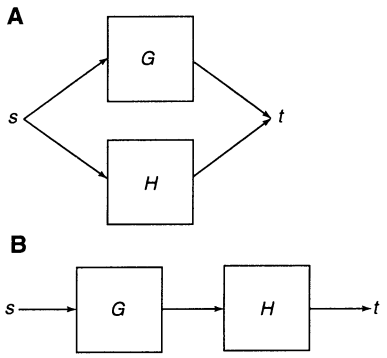

Fig. 3. The networks for formulas $E \vee F$ and $E \wedge F$. Here G is the network for E and H is that for F. (**A**) The OR case is constructed by connecting the networks in parallel and (**B**) the AND case is built by placing the networks in series.

This process assumes that the operations are perfect, that the operations are performed without error. This definitely needs to be studied. The assumption that the extract gets all of the sequences is not needed. If the original test tube has many copies of the desired sequence, then all that is necessary is a reasonable probability that it is correctly extracted to make everything work properly.

These methods can be used to solve a generalization of SAT. This generalization includes most examples of NP problems. The key is to generalize the class of Boolean formulas that we consider. Recall that a SAT problem corresponds to a formula of the restricted form

$$C_1 \wedge \ldots \wedge C_m \tag{6}$$

A natural generalization of this is to consider problems that correspond to any Boolean formula. Thus, we allow formulas to be unrestricted: They can use the logical operations of negation, OR, and AND without any restrictions. More precisely, formulas are defined by the recursive definition

1) Any variable x is a formula.

2) If F is a formula, then so is $\bar{F}$.

3) If F_1 and F_2 are formulas, then so are $F_1 \wedge F_2$ and $F_1 \vee F_2$.

The size of a formula is measured by the number of operations used to build the formula. The SAT problem for formulas is, given a formula F, find an assignment of Boolean values to the variables so that F is true. Because this problem includes normal SAT, it is still NP-complete.

This SAT problem for formulas can be solved in a number of DNA experiments that are linear in the size of the formula. The key to proving this statement is to actually prove more; I show how to solve not just any formula, but any contact net-

work (5). A contact network is a directed graph with a single special source s and a single special sink t. Each edge is labeled with either x or $\bar{x}$, where x is some variable. Given any assignment of values to the variables, an edge is considered to be connected if the edge's formula evaluates to 1. Thus, if the edge is labeled with $\bar{x}$, then it is connected only if $x = 0$. Therefore, the network in Fig. 2 is equal to 1 only if $w = 1$ or $x = y = z = 1$.

The SAT problem for contact networks is to determine whether or not there is an assignment of values to the variables such that there is a directed connected path from s to t. If two edges have the same label, then one is connected if and only if the other is. Put another way, all values of x or $\bar{x}$ are consistent. Our result follows from two simple claims: (i) Given any formula of size S, there is a contact network of size linear in S such that the set of assignments that satisfy the formula also satisfy the network. (ii) Given any contact network of size S, the SAT problem for the network can be solved in order S DNA experiments. These two claims will prove our assertion about formulas.

The first claim is classic (5). Two formulas are equivalent if they always give the same value for any assignment to the variables. Any formula can be placed into a normal form with DeMorgan's Laws

$$\overline{x \vee y} = \bar{x} \wedge \bar{y} \quad (7)$$

$$\overline{x \wedge y} = \bar{x} \vee \bar{y} \quad (8)$$

Through these identities, any formula is equivalent to one where all the negations are on variables. Assuming that our formulas are so restructured, I build a contact network that simulates the formula inductively. If the formula is a variable or its negation, then there is a single-edge contact network that is equivalent. For example, the formula $\bar{x}$ is equivalent to the network with an edge from s to t with the label $\bar{x}$.

In the general case, the formula is equal to either $E \vee F$ or $E \wedge F$, where E and F are simpler formulas. Assuming that G is the network for E and that H is the one for F, the network for $E \vee F$ is constructed by placing G and H in parallel (Fig. 3A). Clearly, there is a connected path from s to t provided that there is either a path from s to t through G or through H. The network for $E \wedge F$ is constructed by placing them in series (Fig. 3B). In this case, there is a connected path from s to t provided there is one through both G and H.

It is quite simple to show how DNA experiments can be used to solve the SAT problem for any contact network. Associate a test tube P_v with each node v in the contact network. The DNA in each test tube should encode in the usual way the set of assignments to the variables that connect s to v. The test tube P_t associated with the sink t is the "answer." Suppose that $v \rightarrow u$ is an edge with the label x ($\bar{x}$) and that P_v is already constructed. Then, construct P_u simply by doing the extraction $E(P_v, x, 1)$ [$E(P_v, x, 0)$]. If several edges leave a vertex v, then use an amplify step to get multiple copies of the DNA in test tube P_v. Also, if several edges enter a vertex v, then pour the resulting test tubes together to form P_v.

The main open question is, of course, if one can actually build DNA computers based on the methods described here. The key issue is errors. The operations are not perfect. I expect that in the near future, experiments will be performed that will determine whether or not DNA-based computers are a practical means of solving hard problems.

REFERENCES AND NOTES

1. L. M. Adleman, *Science* **266**, 1021 (1994).
2. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).
3. R. J. Lipton, "Speeding Up Computations via Molecular Biology," unpublished manuscript (1994) available on the World Wide Web at

   http://www.cs.princeton.edu/~rjl/

   An earlier version of this paper claimed that these results held for circuits as well as formulas. This notion was incorrect. In joint work with J. Sgall, the case for circuits has been solved. The solution for circuits uses additional biological steps.
4. R. R. Sinden, *DNA Structure and Function* (Academic Press, New York, 1994).
5. C. Shannon, *Bell Syst. Tech. J.* **28**, 59 (1949).

6. I would like to thank D. Dobkin for a number of helpful conversations about this work. I would also like to thank L. Adleman for taking the time to explain his wonderful construction. I would also like to thank D. Boneh and C. Dunworth for their help. Finally, the work was supported in part by NSF grant CCR-9304718.

24 January 1995; accepted 30 March 1995

Computation Beyond the Turing Limit

Hava T. Siegelmann

Extensive efforts have been made to prove the Church-Turing thesis, which suggests that all realizable dynamical and physical systems cannot be more powerful than classical models of computation. A simply described but highly chaotic dynamical system called the analog shift map is presented here, which has computational power beyond the Turing limit (super-Turing); it computes exactly like neural networks and analog machines. This dynamical system is conjectured to describe natural physical phenomena.

Humanity's intellectual quest to decipher nature and to master it has led to numerous efforts to build machines—endowed with artificial intelligence—that simulate the world or communicate with it (*1–4*). The computational power and dynamic behavior of such computers is a central question for mathematicians, computer scientists, and physicists. Computer models are ultimately based on idealized physical systems, called "realizable" or "natural" models. Since 1936, the standard accepted model of universal computation has been the Turing machine (*5*), which forms the basis of modern computer science. The Church-Turing thesis, the prevailing paradigm in computer science, states that no realizable computing device can be more powerful (aside from relative polynomial speedups that are a result of richer instruction sets or parallel computation) than a Turing machine (*5*). This report questions that assumption, proposing an alternative model of computation, possibly realizable as well, whose computational power can surpass that of the Turing model. The proposed model builds on a particular chaotic dynamical system (*6*); by applying the system to computer science, a "super-Turing" model can be developed (*7*).

Demonstrating the existence of an ideally realizable super-Turing model has practical and theoretical implications. Theoretically, it could open the way for theories of computation that go beyond the Turing model. On a practical level, computers designed and built on the basis of super-Turing theories should be capable of modeling phenomena that existing computers are not powerful enough to model well.

In computer science, machines are classified according to the classes of tasks they can execute or the functions they can perform. The most popular model is the Turing machine, introduced by the English mathematician Alan Turing in 1936. The Turing machine (*5*) allows for unbounded "external" storage (tapes) in addition to the finite information represented by the current "internal" state (control) of the system. At

Department of Information Systems Engineering, Faculty of Industrial Engineering, Technion, Haifa 32000, Israel. E-mail: iehava@ie.technion.ac.il