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

every step, the machine reads the tape symbol α (where $\alpha \in \{0, 1, \text{blank}\}$) from under an access head, checks the state of the controls (where $s \in \{1, 2, \ldots |S|\}$), and executes three operations:

1) Writes a new binary symbol under the head ($\beta \in \{0, 1\}$).

2) Moves the head one letter to either right or left ($m \in \{L, R\}$).

3) Changes the state of the control ($s' \in \{1, 2, \ldots |S|\}$).

The transition of the machine can be summarized by a function

$$f: (\alpha,s) \rightarrow (\beta,m,s')$$

When the control reaches a special state, called the "halting state," the machine stops. The input-output map (or the function) computed by a Turing machine is defined by the pair of finite binary sequences on its tape before and after the computation, respectively. The finiteness of the input and output is a crucial requirement in computer science, assuring that the ability to compute depends purely on the inherent computational power of the model rather than on a larger amount of external information.

There are many variants of the basic model that yield the same power, such as adding tapes, access heads, and tape dimensions. Other variants may result in more powerful, though nonrealizable, models. Nonuniform Turing machines exemplify such models (*8*): The machines receive on their tape, in addition to the input, another sequence w_n to assist in the computation. For all possible inputs of the same length n, the machine receives the same advice sequence w_n, but different advice is provided for input sequences of different lengths. We are focusing on the class of nonuniform machines that compute in polynomial time (and use a polynomial long advice), denoted by P/poly (*8*). For example, a super-Turing function that appears in P/poly is the "unary halting" function (*5*): Given a unary encoding of a computer program f and a sequence $x \in \{1\}^*$, the function determines whether the program terminates when acting on the sequence x. If both advice and time are exponentially long [that is, $O(2^n)$], the advice can be used to indicate the desired response for each of the 2^n possible input strings of length n and thus to compute all functions f: $\{0, 1\}^* \rightarrow \{0, 1\}$, including noncomputable ones.

Siegelmann and Sontag (*9*, *10*) noticed that such nonuniform classes are indeed very natural for analog computation models. They introduced the first model of computation that is uniform but yet has nonuniform super-Turing capabilities; this model is the classical analog recurrent neural network (ARNN), which is popular in practice as a machine with automatic learning and adaptation capabilities (*11*); see Fig. 1. [A related model is the model of computation over the real numbers (*12*). This interesting model, however, is incomparable with the modern theory of computability, as it does not comply with the finiteness of input and output.] The ARNN consists of a finite number of neurons. The activation of each processor $i = 1, \ldots, N$ is updated by the equation

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}x_j(t) + \sum_{j=1}^{M} b_{ij}u_j(t) + c_i\right) \quad (1)$$

where N is the number of neurons, M is the number of external input signals, x_j are the activations of the neurons, u_j are the external inputs, and a_{ij}, b_{ij}, and c_i are the real coefficients, also called constants or weights (the name "analog" indicates real, rather than rational, coefficients). The function σ is the simplest possible "sigmoid," namely the saturated-linear function

$$\sigma(x) := \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1 \end{cases} \quad (2)$$

(but various other sigmoidal functions may substitute it). A subset of the N neurons is singled out to communicate the output of the network to the environment. Inputs and outputs are streams of letters, and computability is defined under the convention that is sometimes used in practical communication networks: There are two binary input channels, where one is used to carry the binary input signal and the other one indicates when the input is active. A similar convention is applied to the output.

The ARNN computes the super-Turing class P/poly in polynomial time and all binary functions in exponential time (*9*). This fact is connected to classical computability by the observation that when the real weights are constrained to be rational numbers, the network has Turing power only (*10*, *13*). [Follow-up generalizations appear in (*14*–*16*)]. Furthermore, deterministic and probabilistic ARNNs are computationally equivalent (*16*).

Analogous to the Church-Turing thesis, the ARNN was offered as a basic analog computation model, with the conjecture that "any realizable analog computer will have no more power (up to polynomial time) than the analog recurrent networks" (*9*). Here, I assert that although the Church-Turing thesis is indeed a fundamental observation for the large class of discrete computing devices, it may not provide the whole picture, and the analog computation thesis is thus required. To support this assertion, I present a chaotic,

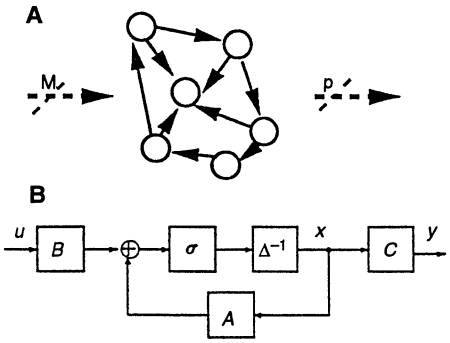


Fig. 1. The analog recurrent neural network (ARNN). (**A**) Graphic view. (**B**) Engineering view.

realizable dynamical system that computationally is as strong as the analog model—the "analog shift map."

In the literature of dynamical systems, chaos is commonly exemplified by the "shift map" [such as the Baker's map (*17*) or the horseshoe map (*18*); Fig. 2] over a set of bi-infinite dotted sequences. Let E be a finite alphabet; a dotted sequence over E (denoted by $\dot{E}$) is a sequence of letters where exactly one is the dot sign (.) and the rest are all in E. The dotted sequences can be finite, (one-side) infinite, or bi-infinite over E. Let $k \in N$ be an integer, the shift map

$$S^k : \dot{E} \rightarrow \dot{E} : (a)_i \rightarrow (a)_{i+k}$$

shifts the dot k places, where negative values cause a shift to the left and positive ones a shift to the right.

I define the "analog shift" as follows: A dotted substring is replaced with another dotted substring (of equal length) according to a function G; then, this new sequence is shifted an integer number of places left or right according to a function F. Formally, the analog shift is the map

$$\Phi : a \rightarrow s^{F(a)}(a \oplus G(a)) \quad (3)$$

where the function

$$G : \dot{E} \rightarrow \dot{E}$$

describes the modification of the sequence, and the function

$$F : \dot{E} \rightarrow \mathbb{Z}$$

indicates the amount of shifting of the dot. Both F and G have a finite domain of dependence (DoD)—that is, F and G depend only on a finite dotted substring of the sequence on which they act. The domain of effect (DoE) of G may be finite, infinite, or bi-infinite. Finally, the operation $\oplus$ is defined by

$$(a \oplus g)_i = \begin{cases} g_i & \text{if } g_i \in E \\ a_i & \text{if } g_i = \epsilon \end{cases}$$

where ϵ is the empty element not contained in E.

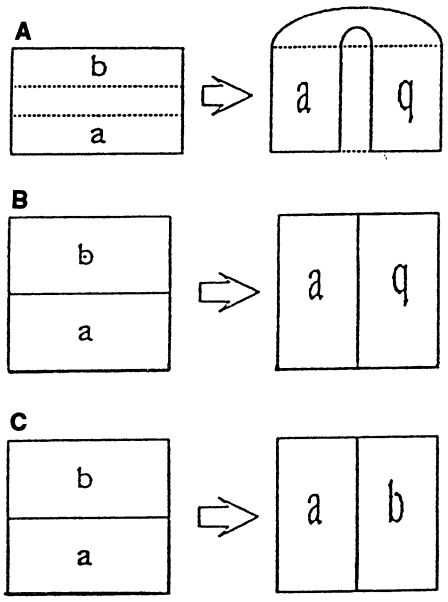


Fig. 2. Various physical shift maps. (**A**) Smale's horseshoe. (**B**) Smale's horseshoe abbreviated to the unit square. (**C**) The Baker's map.

To illustrate, let us assume the analog shift is defined by

DoD	F	G
0.0	1	$\overline{\pi}$.
0.1	1	.10
1.0	0	1.0
1.1	1	.0

Here, $\overline{\pi}$ denotes the left infinite string . . . 51413 in base 2 rather than in base 10. This table is a short description of the dynamical system, in which the next step depends on the first letter to the right of the dot and the one to its left. If these letters (that is, the DoD) are 0.0, then (according to the first row of the table) the left side of the dot is substituted by $\overline{\pi}$ and the dot moves one place to the right. If the DoD is instead 0.1 (as in the second row of the table), the second letter to the right of the dot becomes 0 and there is a right shift, and so on.

The dynamic evolving from

000001.10110

is as follows: here, the DoD is 1.1, hence (by the fourth row of the table) the letter to the right of the dot becomes 0 and the dot is shifted right:

1.1: (000001.00110) 0000010.0110

Now, the DoD is 0.0:

0.0: ($\overline{\pi}$.0110) $\overline{\pi}$0.100

0.1: ($\overline{\pi}$0.100) $\overline{\pi}$01.00

1.0: ($\overline{\pi}$1.00) $\overline{\pi}$01.00

Here, the DoD is 1.0 and no changes occur; this is a fixed point. A special case, when the DoE of G is finite, is the generalized shift map, introduced by Moore (*19, 20*), which was proven to be computationally equivalent to a Turing machine. When G is not applied, we are back to shift maps (*17, 18*).

The computation associated with the analog shift systems is the evolution of the initial dotted sequence until a fixed point is reached, from which the system does not evolve anymore. The computation does not always end; when it does, the input-output map is defined as the transformation from the initial dotted sequence to the final subsequence to the right of the dot. (In the example above, a fixed point is reached in four steps, and the computation was from 000001.10110 to 00.) To comply with the computational constraints of finite input-output, attention was constrained to systems that start with finite dotted sequences and that stop with either finite or left infinite dotted sequences only. Even under these constraints, the analog shift computes richer maps than the Turing machines.

Let AS(k) denote the class of functions computed by the analog shift (AS) map in time k, NN(k) the class of functions computed by the ARNN in time k, and poly(k) the class of polynomials in k; my theorem states that

Theorem 1. *Let F be a function so that $F(n) \geq n$. Then,*

$$AS(F(n)) \subseteq NN(poly(F(n))), \text{ and}$$
$$NN(F(n)) \subseteq AS(poly(F(n))).$$

[Although the result is mathematically profound, a detailed proof is not provided here. For rigorous mathematical proof, see (*21*).]

Sketch of proof. Assume, without loss of generality, that the finite alphabet E is binary; that is, $E = \{0, 1\}$.

1) AS($F(n)$) $\subseteq$ NN(Poly($F(n)$)):
Given a dotted sequence $a = \ldots a_{-3}a_{-2}a_{-1} \cdot a_1a_2a_3 \ldots$, it is mapped into the two sequences

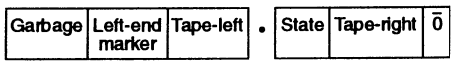
$$a_l = a_{-1}a_{-2}a_{-3} \ldots a_r = a_1a_2a_3 \ldots$$

The analog shift map is redefined as

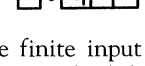
$$\tilde{\phi}(a_l,a_r) =$$
$$(a_l \oplus G_l(d_l,d_r), a_r \oplus G_r(d_l,d_r))$$

where d_l and d_r are the left and right parts, respectively, of the DoD. The value of a_l, as well as a_r, is encoded in a neuron, using a Cantor set representation (*10*). The next operation is decided by the first few bits of both neurons (DoD). The operation involves either substitution of the first finitely many bits (when G is finite to this side) or loading of the neuron with a new value (when G is infinite). These operations require constant time, resulting in an efficient simulation.

2) NN($F(n)$) $\subseteq$ AS(poly ($F(n)$)):
For simulating a Turing machine with advice by an analog shift, its configuration is encoded in a dotted sequence using the fields



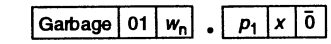
The string starts with an infinite sequence over E (which is mainly ignored), followed by the left-end marker ("01"). Next comes the part of the tape to the left of the read-write head (the pair "10" represents 0 and "11" represents 1), then the dot, the internal state of the machine (encoded as a sequence of 0's ending with 1), the letter under the head and the right part of the tape, and infinitely many 0's that encode the empty part of the tape. Note the asymetry of the left and right empty parts of the tape; this is a result of the asymetric definition of the output of the analog shift. The initial dotted string is



where x is the finite input string. The G function substitutes the left side of the dot with the infinite string

$$w = <\ldots,w_3,w_2w_1>$$

which in polynomially many steps becomes the associated advice (along with lots of garbage)



From this point, each step of the machine is simulated simply; the details are left to the reader.

The appeal of the analog shift map is not only as an almost classical, chaotic dynamical system that is associated with analog computation models. It is also a mathematical formulation that seems to describe idealized physical phenomena. The idealization allows for model assumptions such as any convenient scale to describe the system, noise-free environment, and physics of continuous medium. Some of the physical models, previously used to simulate Turing machines, turn out to be exactly as powerful as the analog models (both to simulate and to be simulated by).

This assertion can be demonstrated, for example, with the system introduced by Moore (*20*). This is a "toy model" describing the motion of a particle in a three-dimensional potential, such as a billiard ball or a particle bouncing among parabolic mirrors. (Note that this system describes a topologi-

cal dynamics rather than a smooth behavior.) A finite number of mirrors suffices to describe the full dynamics, one for each choice of the DoD. The (x,y) coordinates of the particle when passing through a fixed, imaginary plane simulate the dotted sequence $x.y$. To define the computation, the particle starts in input location $\bar{0}.y_o$, where y_o is the finite input string; the output is defined in finite terms as well. Although Moore proved the Turing machine simulation by such a system, the advice can also be encoded in a uniform manner by the characterizations of the mirrors. For example, the concatenation of all advice can be the characterization of the first mirror that is being hit, continuing with mirrors of finite characterizations, simulating the finite DoE. When the halting state of the Turing machine is reached, the particle hits a mirror that throws it to a particular observable x coordinate, where all points are fixed. The output is defined as the y coordinate when this observable x_o is reached. Forcing the input and output to reside in observable areas (using, for example, Cantor set encoding) makes the system realizable. Another possible realization may be based on the recent optical realization of the Baker's map (*22*).

Although it may have seemed that infinite precision was required to fully describe the associated computation, this is not the case, because linear precision suffices for analog computation models (*9*). That is, if one is interested in computing up to time q, both the mirror system and the location of the particle bouncing there are not required to be described (or measured) with more than q bits. This property is in accordance with the sensitivity of chaotic systems to exponentially precise initial conditions (here, the mirror system), which suggests that the analog shift map is indeed a natural model of chaotic (idealized) physical dynamics.

REFERENCES AND NOTES

1. A. M. Turing, *Proc. London Math. Soc (2)* **42**, 230 (1936).
2. W. S. McCulloch and W. Pitts, *Bull. Math. Biophys.* **5**, 115 (1943).
3. C. E. Shannon, in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. (Princeton Univ. Press, Princeton, NJ, 1956), pp. 156–165.
4. J. von Neumann, *ibid.*, pp. 43–98.
5. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Redwood City, CA, 1979).
6. R. L. Devaney, in *Proceedings of Symposia in Applied Mathematics* (American Mathematical Society, Providence, RI, 1989), pp. 1–24.
7. Here, the term "super-Turing" is meant to denote any system of computing that incorporates, but is more powerful than, the standard Turing model. For further discussion of super-Turing Machines, see (*9*).
8. J. L. Balcázar, J. Díaz, J. Gabarr*Structural Complexity* (Springer-Verlag EATCS Monographs, Berlin, 1988–1990), vols. I and II.
9. H. T. Siegelmann and E. D. Sontag, *Theor. Comput. Sci.* **131**, 331 (1994).
10. ———, *J. Comput. Syst. Sci.*, in press. A previous version of this appeared in the *Proceedings of the Fifth ACM Workshop on Computational Learning Theory*, Pittsburgh, PA, July 1992.
11. J. Hertz, A., Krogh, R. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Redwood City, CA, 1991).
12. L. Blum, M. Shub, S. Smale, *Bull. Am. Math. Soc.* **21**, (1989).
13. H. T. Siegelmann and E. D. Sontag, *Appl. Math. Lett.* **4**, 77 (1991).
14. P. Koiran, M. Cosnard, M. Garzon, *Theor. Comput. Sci.* **132**, 113 (1994).
15. J. L. Balcázar, R. Gavaldà, H. T. Siegelmann, E. D. Sontag, *Proceedings of the IEEE Structure in Complexity Theory Conference*, San Diego, CA, May 1993, pp. 253–265.
16. H. T. Siegelmann, in *Lecture Notes in Computer Science, 820: Automata, Languages and Programming*, S. Abitebul and E. Shamir, Eds. (Springer-Verlag, Jerusalem, 1994).
17. V. I. Arnold and A. Avez, *Ergodic Problems of Classical Mechanics* (Benjamin, New York, 1986).
18. J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamics Systems, and Bifurcations of Vector Fields* (Springer-Verlag, New York, 1983).
19. C. Moore, *Nonlinearity* **4**, 199 (1991).
20. ———, *Phys. Rev. Lett.* **64**, 2354 (1990).
21. H. T. Siegelmann, "The simple dynamics of super Turing theories" (Technical Report 94-NN-1, Technion, 1994).
22. J. P. Keating Hannay, J. H. Dealmeida, A. M. O. Dealmeida, *Nonlinearity* **7**(5), 1327 (1994).

23. I thank A. Ponak (University of Calgary), J. Schiff (Bar-Ilan University), and S. Fishman (Technion) for helpful comments.

26 July 1994; accepted 25 January 1995

Dating and Context of Three Middle Stone Age Sites with Bone Points in the Upper Semliki Valley, Zaire

Alison S. Brooks, David M. Helgren, Jon S. Cramer, Alan Franklin, William Hornyak, Jody M. Keating, Richard G. Klein, William J. Rink, Henry Schwarcz, J. N. Leith Smith, Kathlyn Stewart, Nancy E. Todd, Jacques Verniers, John E. Yellen

The extent to which the earliest anatomically modern humans in Africa exhibited behavioral and cognitive traits typical of *Homo sapiens sapiens* is controversial. In eastern Zaire, archaeological sites with bone points have yielded dates older than 89^{+22}_{-15} thousand years ago by several techniques. These include electron spin resonance, thermoluminescence, optically stimulated luminescence, uranium series, and amino acid racemization. Faunal and stratigraphic data are consistent with this age.

During the late middle to early upper Pleistocene in Africa, anatomically modern humans (*Homo sapiens sapiens*) replaced archaic *Homo sapiens*. Middle Stone Age (MSA) archaeological materials provide information on human behavior during this transition. In tropical Africa, MSA artifacts with associated fauna and chronometric ages are known from only a few well-excavated rock-shelter and stratified open-air contexts (*1*, *2*). In this report, we describe the geological context and dating of three MSA sites with bone points in eastern Zaire.

The Semliki Valley (Fig. 1) runs north-northeast along the floor of the western (Albertine) branch of Africa's modern rift valley system, from Lake Rutanzige (formerly known as Lake Edward) to Lake Mutanzige (formerly Lake Albert) (*3*). Sites along the northern shore of Lake Rutanzige and the Upper Semliki Valley range from Pliocene to Holocene age (*4*).

The current savanna-woodland vegetation and fauna of the Upper Semliki Valley are a response to rain-shadow microclimatic effects of the western rift wall and to the porous, base-rich ash of early Holocene age that blankets the local landscape (*5*). These factors may have been quite different during the Pliocene and Pleistocene. In contrast to the eastern (Gregory) rift, exposures of Plio-Pleistocene and Pleistocene sediments in the Semliki are very limited. The chronology of the sequence before this study was based largely on faunal comparisons and lithostratigraphy.

Early archaeological work (*4*, *6*) focused primarily on the lake-shore site of Ishango, with its small barbed bone and ivory points, fish and mammal bones, fragmentary human remains, quartz tools, and an engraved bone haft that may indicate an understanding of multiplication by 2's. Ishango and other archaeological and paleontological occurrences (*7*) attracted renewed multidisciplinary research in the Upper Semliki (*8*–*12*) between 1982 and 1990.

New materials (Table 1), especially ostrich eggshell, were recovered for dating from the original Ishango site (Ishango 11) and from a comparable-age site 2 km downstream (Ishango 14) (*13*). Together with a restudy of the original fauna by Peters (*14*), these suggested that the *niveau fossilifère*