COMPUTER RECREATIONS

The cellular automaton offers a model of the world and a world unto itself

by Brian Hayes

It is due cause for wonder that molecules of water "know" how to frame the elaborate symmetries of a snowflake. There is no architect directing the assembly, and the molecules themselves carry within them no template for the crystalline form. Pattern on a large scale emerges entirely from the short-range interactions of many identical units. Each molecule responds only to the influence of its nearest neighbors, but a consistent arrangement is maintained throughout a structure made up of perhaps 10^{20} molecules.

One way to approach an understanding of this process is to imagine that each site where a molecule might be emplaced is governed by a rudimentary computer. As the crystal grows, each computer surveys the surrounding sites and, depending on its findings, determines by some fixed rule whether its own site should be occupied or vacant. The same calculation is made at all the sites according to the same rule.

The computational model of snowflake growth is a cellular automaton: a uniform array of many identical cells, or sites, in which each cell has only a few possible states and interacts only with a few neighboring cells. The components of the system—the cells and the rule for calculating the next state of a cell—can be simple indeed and nonetheless give rise to a remarkably complex evolution.

The idea of the cellular automaton is roughly as old as the electronic digital computer. The first investigations were carried out by John von Neumann (with an important contribution from Stanislaw Ulam) in the early 1950's. Von Neumann's primary aim was to devise a simple system capable of reproducing itself in the manner of a living organism. The best-known cellular automaton, the "game of life" invented in 1970 by John Horton Conway, also has a biological aspect, as the name suggests; cells are born, live or die depending on the local population density.

In more recent work on cellular automata the emphasis has shifted somewhat. Arrays of locally interacting cells are seen as potentially useful models of physical systems, ranging from snowflakes to ferromagnets to galaxies. They may also have applications to questions in computer science, both practical (How should one organize a network of many interacting computers?) and theoretical (What is the ultimate limit to the power of a computing machine?). Perhaps most intriguing, the cellular automaton can be viewed as a "digital universe" worth exploring for its own sake, quite apart from its utility as a model of the real world.

The resurgence of interest in cellular automata was marked by a workshop on the subject held a year ago at the Los Alamos National Laboratory. The proceedings (some 20 papers) have since been published in *Physica D* and in book form by the North-Holland Publishing Company. Almost all of what is reported here is based on work discussed at the Los Alamos meeting.

Four properties characterize a cellular automaton. The first property is the geometry of the array of cells. For a model of snowflake growth a two-dimensional hexagonal array would be appropriate, but in most contexts a rectilinear lattice is chosen, one made up of identical squares. Arrays with three or more dimensions are readily constructed but are not readily visualized. Lately surprising discoveries have been made with the still simpler one-dimensional array: a mere line of cells.

Within a given array it is necessary to specify the neighborhood that each cell examines in calculating its own next state. In the two-dimensional rectilinear array two neighborhoods have been given much attention. Von Neumann confined each cell's attention to its four nearest neighbors, those to the north, south, east and west; this set of cells is now called the von Neumann neighborhood. The neighborhood that includes these four cells and the four diagonally adjacent ones is called the Moore neighborhood, after Edward F. Moore. Obviously neighborhoods overlap, and a given cell is simultaneously included in the neighborhoods of several adjacent cells. In some cases the center cell—the cell making a calculation—is considered a member of its own neighborhood.

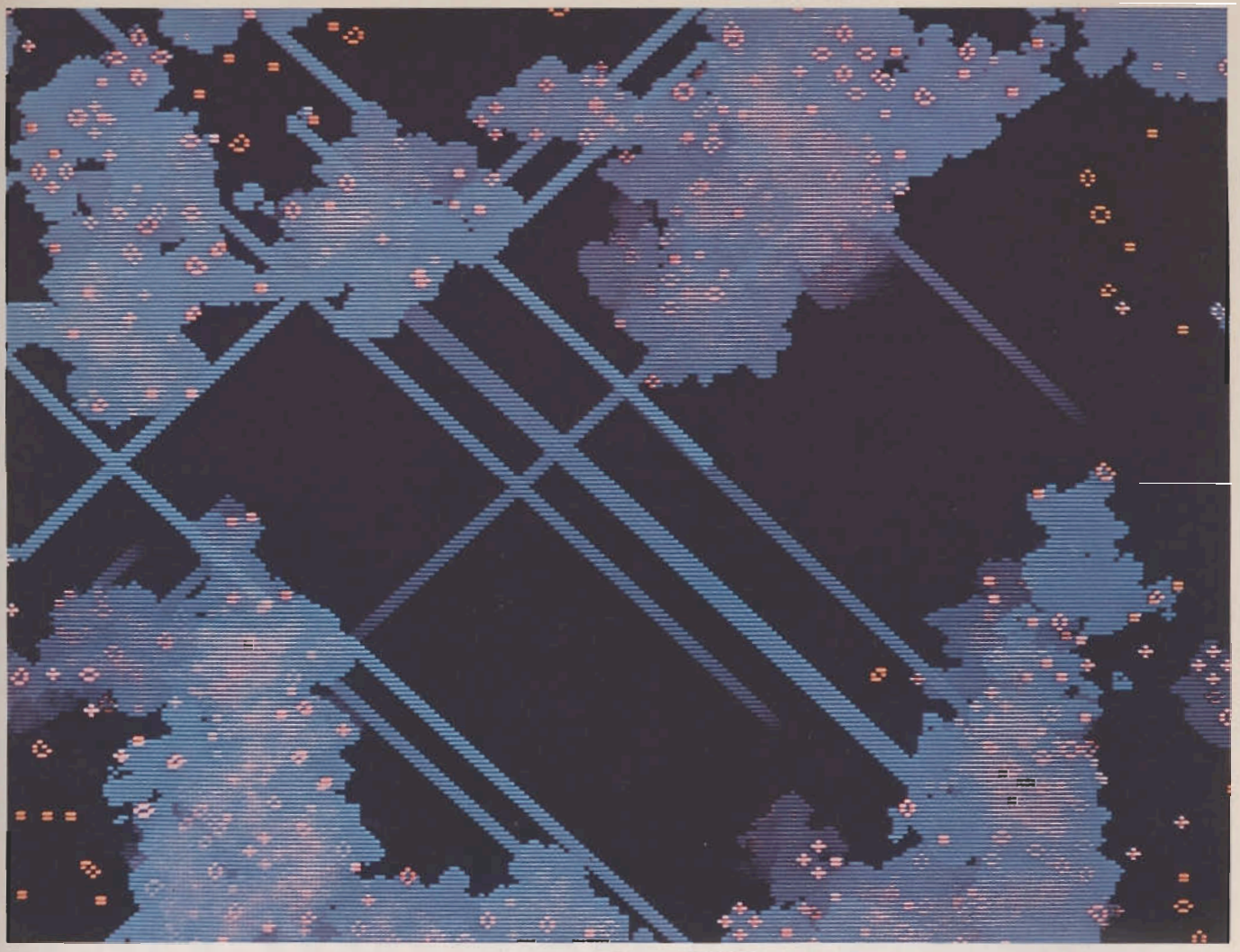
The third factor to be considered in describing a cellular automaton is the number of states per cell. Von Neumann found a self-replicating pattern made up of cells with 29 possible states, but most automata are far simpler. Indeed, there is ample scope for variation even among the binary automata, those with only two states per cell; the states may be represented as 1 or 0, true or false, on or off, living or dead.

The primary source of variety in the universe of cellular automata is the enormous number of possible rules for determining the future state of a cell based on the present configuration of its neighborhood. If k is the number of states per cell and n is the number of cells included in the neighborhood, there are k^{k^n} possible rules. Thus for a binary automaton in the von Neumann neighborhood (where n is 4) there are more than 65,000 possible rules; in the Moore neighborhood (where n is 8) there are 10^{77}. Only a trifling fraction of them have been examined at all.

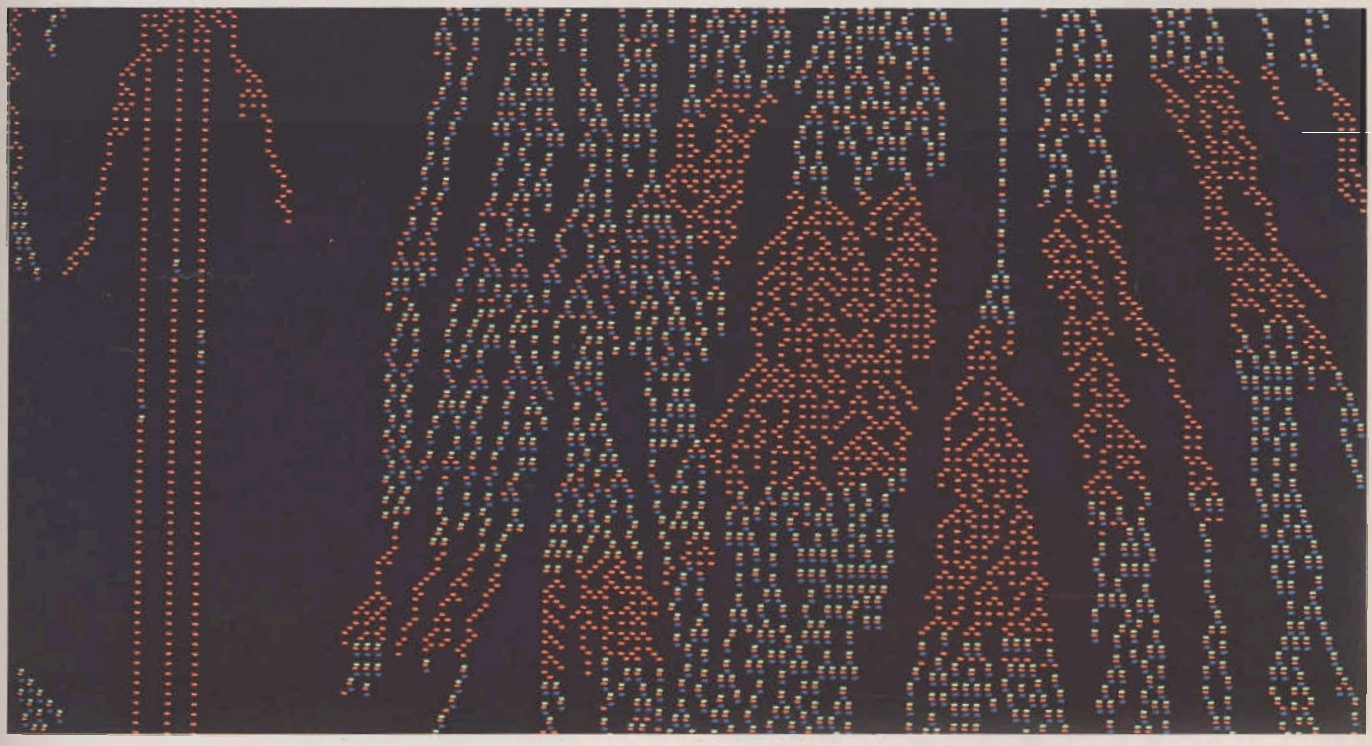
The game of life is played with two-state cells on a rectilinear lattice in the Moore neighborhood, with the additional complication that the center cell is significant. In other words, at each step in the evolution of the system every cell checks the state of the eight surrounding cells as well as its own state. According to the rule defined by Conway, if the center cell is living, it will continue to live in the next generation if either two or three of the eight cells in the neighborhood are also living. If there are three live cells in the neighborhood, the center cell is alive in the next generation regardless of its present state. Under all other circumstances the center cell either dies or remains dead.

The fascination of the game of life is its unpredictability. Some patterns die out entirely; many more lapse into a stable configuration or a cyclical one with a period of a few generations. Over the years, however, a number of more interesting initial states have been discovered, such as the "glider gun" that launches an unending stream of projectiles. The exploration of life's byways continues. Recent developments are described by Martin Gardner in *Wheels, Life, and Other Mathematical Amusements*. Here I should like to turn to other cellular automata whose properties are just beginning to be elucidated.

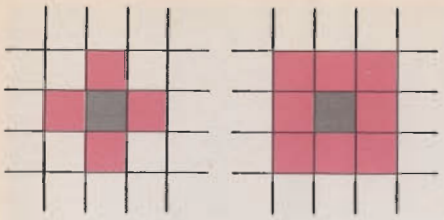
Among the multitude of possible transition rules, many hold little intrinsic interest. For example, a rule stating that a cell will be on if and only if the cell to its left is on specifies an evolution that is quite easy to predict: any initial pat-



The "game of life" evolves on the screen of Tommaso Toffoli's cellular-automaton machine



A pattern of dendrites is created by a cellular automaton with an asymmetric transition rule



The von Neumann and Moore neighborhoods

tern preserves its shape but shifts to the right by one cell with each time step. A subclass of rules called counting rules or totalistic rules seems to include specimens of almost all the observed varieties of cellular automata. With rules of this kind the new state of a cell depends only on the number of neighbors in a given state, not on their position. Many automata based on such rules have been investigated by members of the Information Mechanics Group of the Laboratory for Computer Science at the Massachusetts Institute of Technology. The group consists of Edward Fredkin, Norman Margolus, Tommaso Toffoli and Gérard Y. Vichniac.

One of the simplest counting rules is the parity rule, which assigns a cell a value of 1 if an odd number of the neighboring cells are 1's and otherwise assigns it a value of 0. The evolution of this system, when the rule is applied in the von Neumann neighborhood, was described in this space last October. Any starting pattern is replicated four times; the four copies are then replicated in turn, and so on.

Another class of counting rules are the "voting" rules, which give the center cell a value of 1 whenever the number of 1's in the neighborhood exceeds some threshold. Vichniac, in a paper presented at the Los Alamos meeting, points out that rules of this type yield models of percolation and nucleation, phenomena of importance in solid-state physics and other fields. Percolation is the term applied to the formation of an unbroken path across some space; for example, when a metal is dispersed in an insulating matrix, the conductivity of the composite depends on the probability of forming a continuous chain of metal atoms. Likewise the transmission of an infectious disease is possible only through an unbroken sequence of susceptible individuals. Nucleation is the process that initiates the growth of a crystal, the boiling of a liquid and similar events.

One transition rule that gives rise to percolation makes the center cell a 1 only if there are 1's in at least three out of the five cells that constitute the von Neumann neighborhood plus the center cell. The onset of percolation is extremely sensitive to the initial concentration of 1's. If the concentration is less than one-half, continuous chains of 1's spanning the array are not likely to form in the course of the evolution. At a concentration of one-half or greater the chains do appear, but the entire lattice still does not fill with 1's; islands of 0's remain in the final stable state. Nucleation, in which the array does fill solidly with 1's, is observed when the rule is changed to require only two out of five 1's. The critical concentration is .0822.

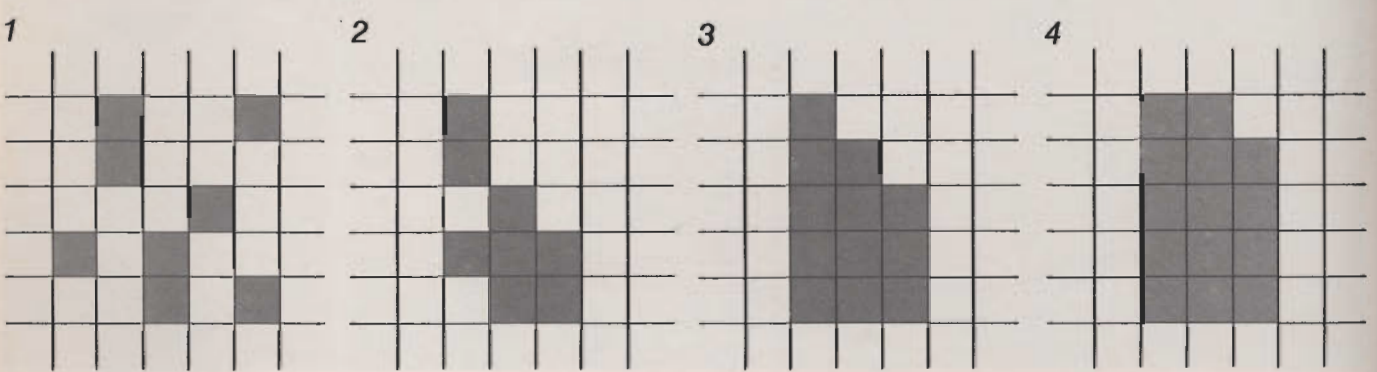
The Ising model is a conceptual tool of physics that seems superficially to be much like a cellular automaton. The model is a rectilinear lattice where each site has two possible values and interacts only with its four nearest neighbors. The model is often employed to describe ferromagnetic materials; each site represents an atomic spin that must point either up or down. Below a critical temperature (the Curie temperature) the spins tend to be aligned, so that the material is magnetized, but at higher temperatures they are more or less randomly distributed.

In October I discussed a version of the Ising model created with a spreadsheet program, whose lattice of cells lends itself naturally to cellular-automata studies, albeit a lattice with probabilistic rules to emulate temperature. I observed a curious phenomenon: at low temperature the spins did not assume a uniform alignment in one direction; instead they adopted a checkerboard configuration of alternating up and down spins. With each time step all the spins flipped. In a ferromagnet the checkerboard pattern is the configuration of highest energy and should therefore be unstable; it is the pattern characteristic of an antiferromagnet.

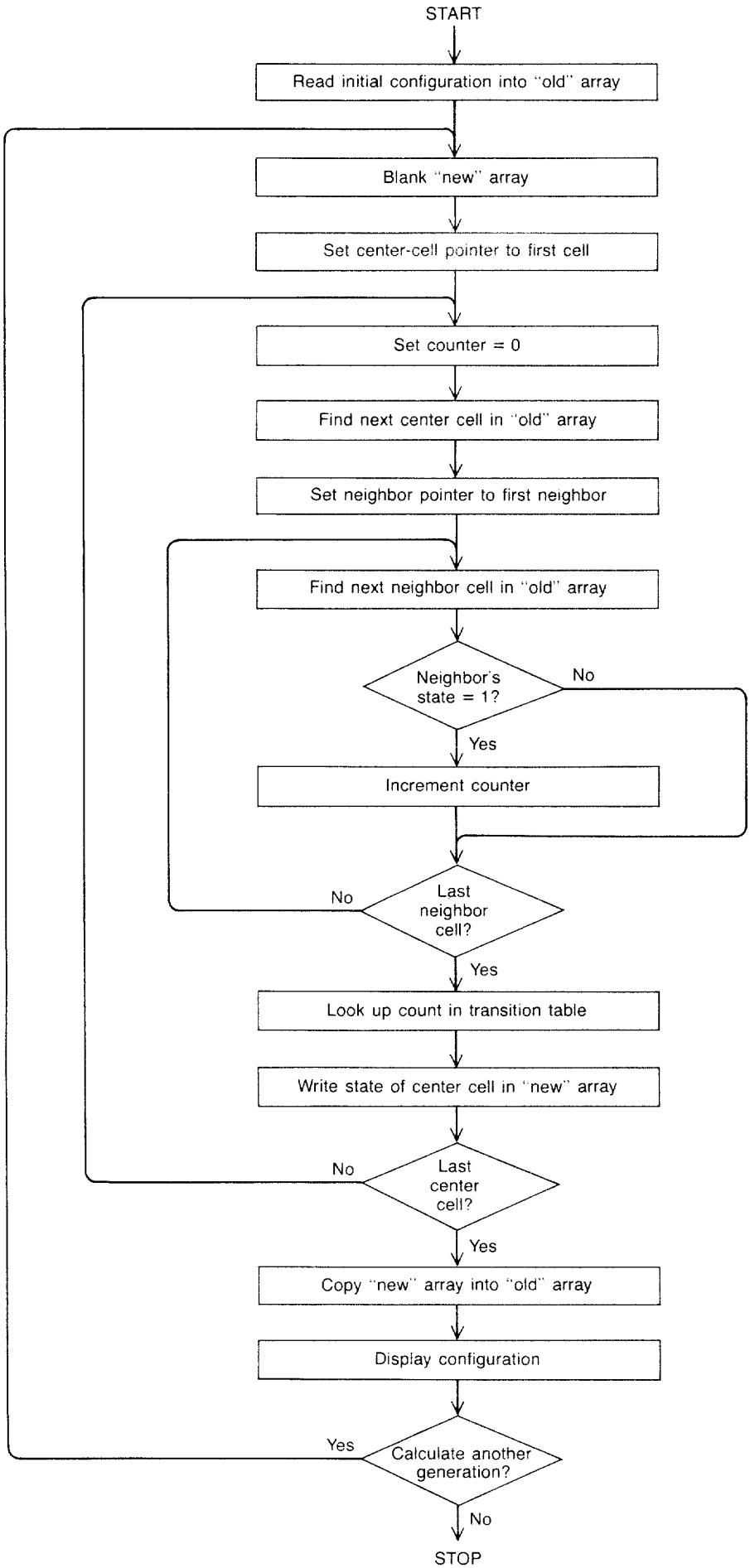
Vichniac had already discovered the problem and explained it. In the standard implementation of the Ising model only one spin is allowed to change in each iteration. It follows that when a particular site surveys its neighborhood, some of the spins it inspects are "old" ones and some are "new." Under these conditions the oscillating antiferromagnet cannot arise. It is only when all the spins are recalculated simultaneously that the high-energy antiferromagnet is favored. There are strategies for avoiding this "feedback catastrophe," but the lesson of larger significance is that the simplest intuitive correspondence between the Ising model and cellular automata is misleading.

Vichniac and others in the M.I.T. group point out that cellular automata have a status fundamentally different from that of other physical models. The commonest device for building a mathematical model of the natural world has long been the differential equation, which can describe the change in some quantity as a function of position and time. For example, Maxwell's equations give the variation in the value of an electromagnetic field from point to point and from moment to moment. All the quantities in such equations are continuous: they vary smoothly. A cellular automaton, on the other hand, is a fully discrete system. Space is not a continuum but an array of cells; time too is broken down into discrete steps, and whereas the magnitude of a field can vary over a continuous range, the cells of a cellular automaton can have only a finite number of states.

Of course, real space and time and many physical variables are thought to be continuous rather than discrete (at least at the scale commonly considered). It does not follow, however, that differential equations yield inherently superior models of nature. Often it is not the precise numerical value of a variable that is significant but only its overall size, as in whether a particular point in



Evolution of a cellular automaton under the two-out-of-five voting rule



Algorithm for cellular automata based on "counting" or "totalistic" transition rules

a growing snowflake is ice or water vapor. Cellular automata make this discreteness explicit on a digital computer. In addition their temporal evolution can be computed exactly; there is no need for approximations. Furthermore, they can make far more efficient use of the digital computer's resources.

A program for simulating a cellular automaton can be written for even the smallest computer. Indeed, Per Bak of the Brookhaven National Laboratory has recently argued in *Physics Today* that many simulations in physics can be done more effectively and more cheaply with a small personal computer than they can be with more powerful shared facilities. The example he chose to illustrate was a simulation of the three-dimensional Ising model, done with a Commodore VIC-20 computer at an estimated cost of $4.

The most straightforward cellular-automaton program simply embodies the method one would be likely to adopt in carrying out the procedure by hand with graph paper. First an array of cells is established, with each cell being represented by a memory element in the computer. For each time step the program must attend to every cell in turn, examine its neighbors and calculate the appropriate value for the cell's next state. The calculation itself is conveniently done by looking up the value in a table. If only counting rules are considered, the table needs only one entry for each possible number of "on" cells. When other kinds of rules are allowed, the table can become quite elaborate.

A few subtleties must be kept in mind when one writes such a program. Most important is the need to avoid altering the content of a cell before its value has been checked by all the other cells to which it is a neighbor. The easiest way of meeting this requirement is to maintain two copies of the array; the program examines one copy to determine the current state of the neighborhood and enters the result of its calculation in the other copy. Boundary conditions must also be defined. Ideally the array would be infinite, but that is clearly impractical. A common technique is to effectively join the edges of an array, so that cells on opposite edges become neighbors. In one dimension an array of this kind is topologically a circle and in two dimensions it is a torus; although it is finite, it has no boundaries.

A program of the kind described above, running on a general-purpose digital computer, is a sequential procedure that simulates the actions of an array of many computers running simultaneously. Far better would be an actual network of multiple computers with the structure of the cellular array. Building such a machine is by no means out of the question: the individual computers

would be so simple that many of them might be fitted onto a single semiconductor chip. The fact that only nearby computers need to communicate with one another would also reduce the complexity of the device. Toffoli has estimated that such a processor might operate faster than a general-purpose computer by a factor of a million or even a billion. Preliminary work on computers of this kind is under way at the Massachusetts Institute of Technology and the Thinking Machines Corporation of Waltham, Mass.

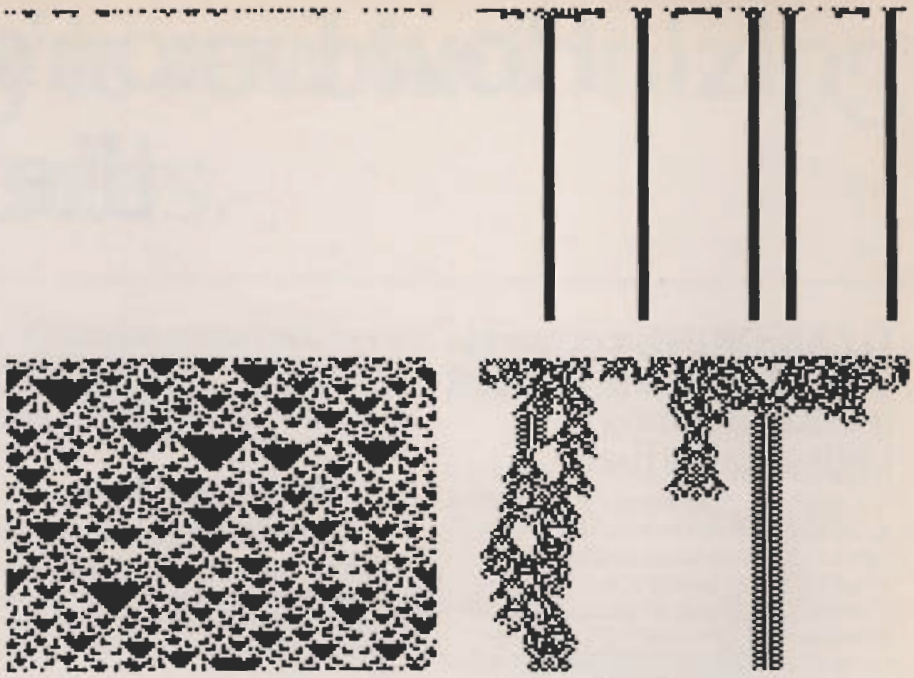
In lieu of a special-purpose chip, Toffoli has constructed a dedicated cellular-automaton machine out of standard microelectronic components. Calculations are done serially rather than for all the cells at once, but because the device is finely tuned to a single kind of calculation, it is roughly 1,000 times faster than a general-purpose computer. The machine itself consists of a few printed-circuit boards mounted in a frame; it is connected to a color display and is controlled by another small computer, an Atari 800.

Toffoli's cellular-automaton machine provides an array of 256 by 256 cells, each of which can have up to 256 states. The state of every cell is recalculated 60 times per second. Watching a system evolve at this rate is quite different from watching a slower device. Instead of a sequence of still photographs one sees a motion picture. The game of life no longer appears as a stately progression of abstract patterns; it is more like a view through the microscope of bacteria and protozoa swimming, spinning, breeding, eating and being eaten.

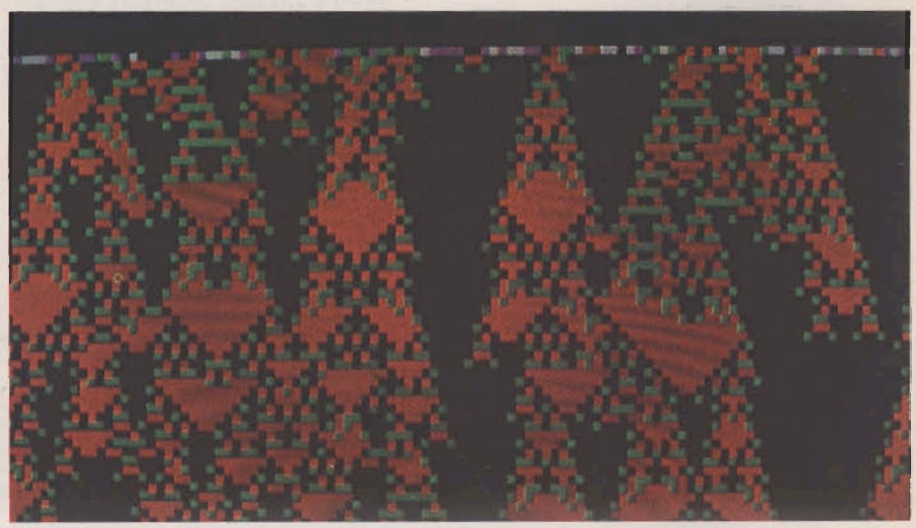
A one-dimensional cellular automaton is much less demanding of computer resources, both spatial and temporal, than a two-dimensional system. Writing a program for the one-dimensional system is also easier. The linear array has still another advantage over the planar one: because of the simpler geometric structure, there is more hope of gaining an analytic understanding of the automaton's evolution. In the past two years Stephen Wolfram of the Institute for Advanced Study has undertaken to do just that.

A single generation of a one-dimensional array is merely a line of cells, but successive generations can be plotted next to one another. In this way a two-dimensional pattern is formed that has one spatial axis and one time axis, and the entire evolution of the system can be taken in at a glance.

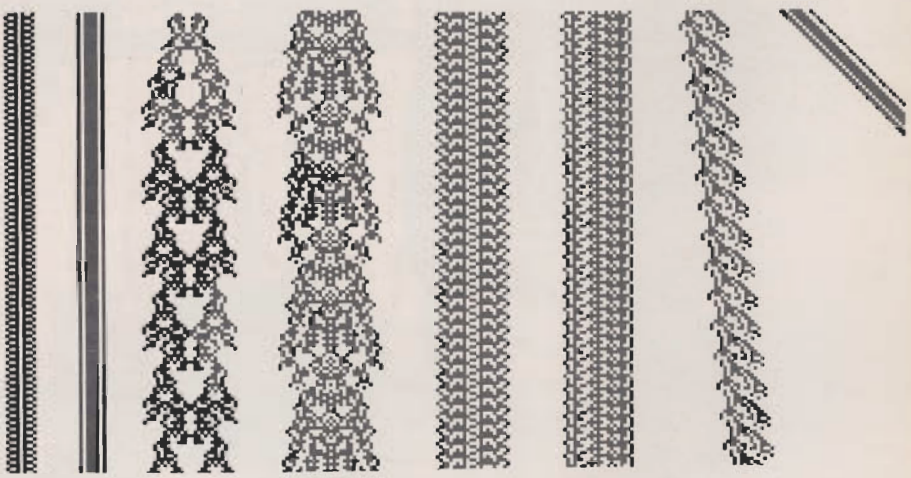
Wolfram has found that all the transition rules he has investigated so far can be put into just four classes. Class 1 consists of those rules whose evolution leads to a stable and homogeneous state; for example, all cells might take on a value of 0 or of 1. Class 2 rules give rise



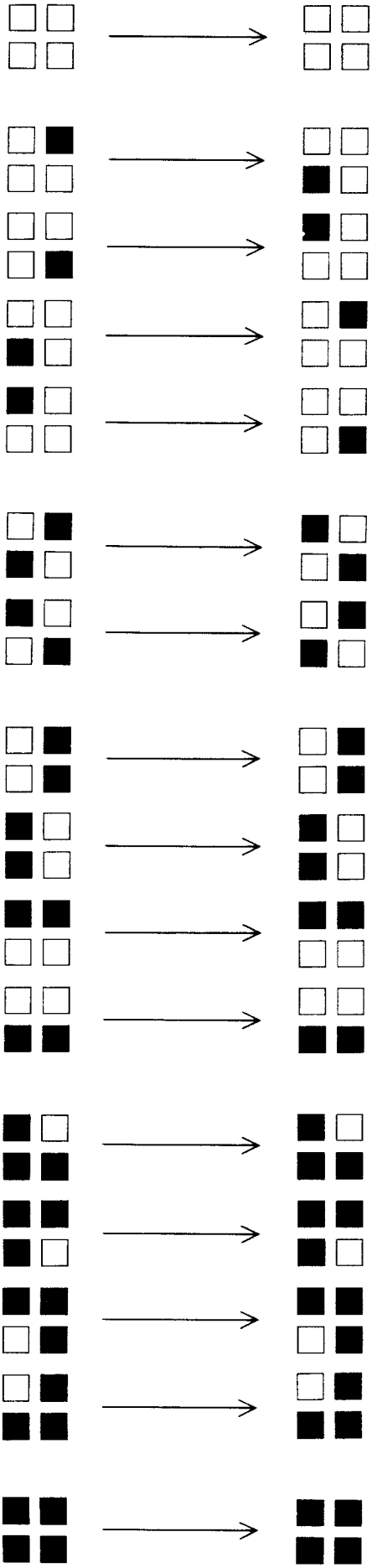
The four classes of totalistic rules in one dimension



Successive states of a Class 4 one-dimensional automaton



Some components of a possible universal computer



Transition rule for a billiard-ball computer

to simple structures that are either stable or periodic but in either case remain isolated from one another. The rules in Class 3 create chaotic patterns, although not random ones. In Class 4 are the few transition rules that generate structures of substantial spatial and temporal complexity.

Wolfram conjectures that the one-dimensional cellular automata may be the simplest well-defined systems capable of complex self-organizing behavior. In nature many continuous dynamical systems have such a capability: beginning in a random initial state, they evolve a highly ordered structure. (The snowflake is an example.) The evolution can be explained in terms of attractors, which seem to draw the system toward a subset of all the possible configurations.

A parallel has been established between the classes of cellular automata and the kinds of attractors observed in physical systems. A Class 1 automaton is analogous to a continuous system with the simplest attractor: a limit point, which invariably brings the system to the same final state. The evolution of a Class 2 automaton is rather like that of a system with a limit cycle, a set of configurations that is repeated indefinitely.

Class 3 automata, with their chaotic patterns, can be associated with the more interesting entities called strange attractors, which are characteristic of physical phenomena such as the onset of turbulent flow. In a system governed by a strange attractor evolution proceeds toward a subset of all the possible configurations, but the subset can have an exceedingly intricate structure. When the set is visualized as an array of points in space, it is in many cases a fractal, a geometric figure with a fractional number of dimensions.

The distinctions between the classes of automata can be made clearer by considering a simple experiment. Suppose a cellular automaton is started in some randomly chosen initial configuration and allowed to evolve for many time steps; the final state is then noted. Now return to the starting configuration, change the value of a single cell and allow the system to evolve for the same number of steps. What effect will the small change have on the final state? In a Class 1 automaton there is no effect at all: a Class 1 system reaches the same final state no matter what the initial state is. A Class 2 automaton may show some effect, but it is confined to a small area near the site of the change. In a Class 3 system, however, altering a single cell can set up a disturbance that propagates throughout the array.

The Class 4 rules are the rarest and the most intriguing. Some quite simple transition functions fall into this class; for example, in the neighborhood defined to include the center cell and the two cells on each side of it, the rule stating that the center cell is a 1 if either two or four cells in the neighborhood are 1's leads to Class 4 patterns. Sensitivity to small variations in the initial conditions is even greater in Class 4 than it is in Class 3. It is conjectured that in predicting the future state of a Class 4 automaton there can be no general procedure more efficient than allowing the automaton itself to compute the state.

A related conjecture has even grander scope: it suggests that Class 4 automata may qualify as universal computers. The Turing machine is the most familiar device of this kind; if a function can be computed at all, a Turing machine can presumably do it. Other computers can be proved universal by showing that they are equivalent to a Turing machine. Several two-dimensional cellular automata (including the game of life) have been shown to be universal computers, and a proof has also been given for a complicated one-dimensional system with 18 states per cell. The Class 4 automata would be the simplest universal computers known. Most of the essential components have been identified. One important missing element is a clock: a structure that issues a train of pulses at regular intervals, like the glider gun in the game of life.

The view of cellular automata as computers suggests that their self-organizing behavior can be characterized in terms of their computational capabilities. Thus, for example, sets of configurations generated by the evolution of a cellular automaton can be thought of as a formal language. Each configuration is considered as a word in the language, formed from a sequence of symbols representing the cellular-automaton site values according to a set of grammatical rules. Wolfram has shown that the configuration generated by any cellular automaton after a finite time can be described by a simple class of formal languages known as regular languages. For any of these regular languages it is possible to find a simplest grammar. That grammar gives a minimal description of the cellular-automaton configurations, and its size can be taken to measure the complexity of the configurations. For cellular automata of Class 1 and Class 2 the complexity tends to a finite limit at large times, so that the structures generated by these systems are described by regular languages. For cellular automata of Class 3 and Class 4, however, the complexity usually increases rapidly with time, and it appears that more complicated formal languages are required to describe the large-time behavior of such systems.

There is a special class of cellular automata that are said to be reversible, or invertible. From any starting configu-

ration a reversible automaton can be allowed to evolve for any number of time steps, then stopped and run in reverse, and it will return to its exact initial state. The patterns formed by a typical reversible automaton have a qualitatively different appearance from those characteristic of a nonreversible automaton. In particular, if the pattern is initially random, it tends to remain random; no self-organizing structures appear.

A necessary condition for reversibility is that the transition rule be deterministic in both the forward and the backward directions, that is, every possible state of a neighborhood must have both a unique successor and a unique predecessor. The game of life is nonreversible because the predecessor of a state cannot be identified unambiguously: if a cell is currently "dead," for example, in the preceding generation it could have had any number of living neighbors other than three. A systematic way of creating reversible transition rules was invented by Fredkin and has been further investigated by Margolus. The essence of the method is to let the next state of a cell depend on the two previous states of the neighborhood. The state at time $t+1$ is given by any function of the neighborhood at time t minus the state at time $t-1$. The reversal is then straightforward: the state at time $t-1$ must be given by the state at time t minus the state at time $t+1$.

Because of the requirement of bidirectional determinism, there can be no attractors in the evolution of a reversible automaton. The presence of an attractor implies that many initial states evolve along paths that merge with one another; in the reversed evolution the merger points would become branch points, where determinism would fail. Similarly, a reversible cellular automaton can never enter or leave a loop, or cycle of states, because again a branch point arises in one direction or the other. Because attractors and the associated self-organizing patterns are excluded, it may seem that reversible transition rules would give rise to quite dull cellular automata, but other features of the systems offer compensating points of interest. Most notably, the information content of a pattern of cells in a reversible automaton turns out to be a conserved quantity (one that cannot increase or decrease in the course of the automaton's evolution). This property makes the reversible systems valuable models of computation.

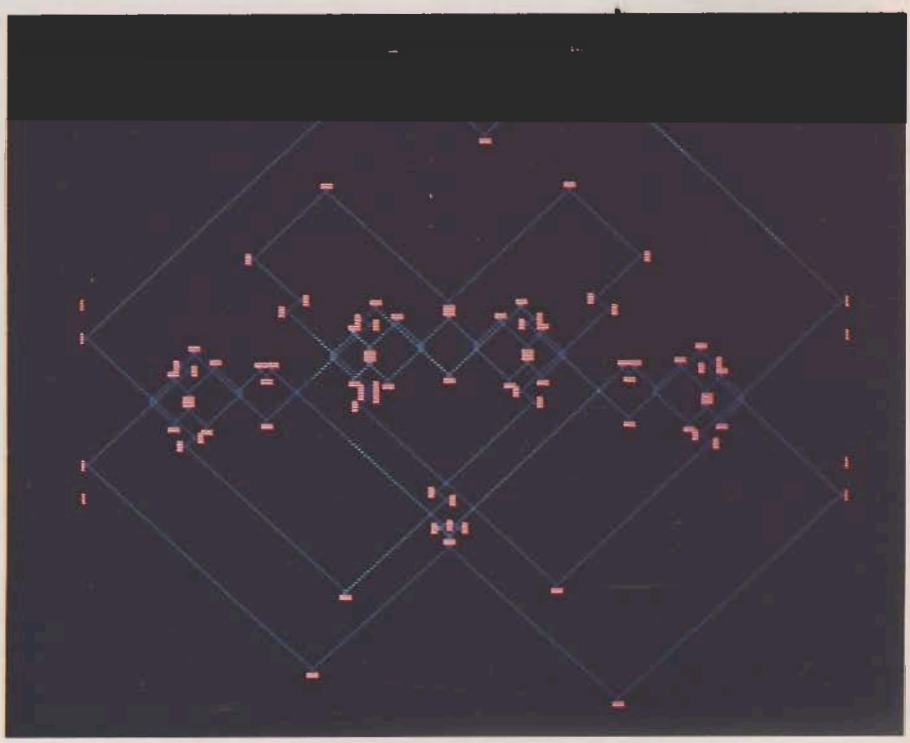
Margolus has constructed a cellular-automaton computer based on an imaginary mechanical system first discussed by Fredkin: the billiard-ball model of computation. In the model bits of information (1's and 0's) are carried by idealized billiard balls that move without friction and rebound from one another and from other obstacles with perfect elasticity. The presence of a ball at a designated position represents a binary 1 and the absence of a ball represents a binary 0. Through a clever arrangement of bumpers it is possible to create various logic gates analogous to those of an electronic computer. In an AND gate, for example, one billiard ball passes through the output region (and thereby registers a binary 1) only if two balls approach the gate simultaneously along specific trajectories.

Margolus' cellular-automaton version of the billiard-ball model is an example of a simple but somewhat unusual reversible transition rule. Cells are considered not individually but in blocks of four; every possible pattern within a block is transformed into a unique product pattern. The rule is designed so that a single 1 in a background of 0's propagates along one of the four diagonal directions of the lattice at a speed of one cell per time step; the isolated 1 is the embodiment of a billiard ball. A solid block of four 1's remains unchanged and acts as a perfect reflector. When the model is set going on Toffoli's cellular-automaton machine, the "billiard balls" streak across the display screen in intricate interwoven patterns. Watching this orderly (if frenetic) motion, it is hard to keep in mind that the program has no representation of the balls' paths but merely applies a single rule to all the cells.

The billiard-ball model and its cellular-automaton implementation have an important bearing on the theory of computation. It has been conjectured that any computer must have components that dissipate both energy and information; according to this argument, there is a thermodynamic limit to the efficiency of a computer just as there is to the efficiency of a heat engine. The supposedly inevitable losses of information and energy result directly from the irreversibility of the computational process. (When a computer adds the numbers 5 and 3 to get 8, the procedure cannot be reversed because there are infinitely many numbers that could have been added to get the same result.)

Fredkin, Toffoli and Margolus point out that the billiard-ball model offers a counterargument to the notion of inevitable dissipation. In the billiard-ball computer no information is lost. Indeed, the billiard balls themselves cannot be created or destroyed, and all the information that defines their initial pattern is preserved as the system evolves. The inputs to an addition operation can be recovered simply by reversing the trajectories. In principle the billiard-ball computer could operate with no internal power consumption.

The connection between physics and computing has been made with particular clarity by Toffoli in a statement that could be read as a description of the largest of all cellular automata. "In a sense," he writes, "nature has been continually computing the 'next state' of the universe for billions of years; all we have to do—and, actually, all we *can* do—is 'hitch a ride' on this huge ongoing computation, and try to discover which parts of it happen to go near to where we want."



The billiard-ball computer in action