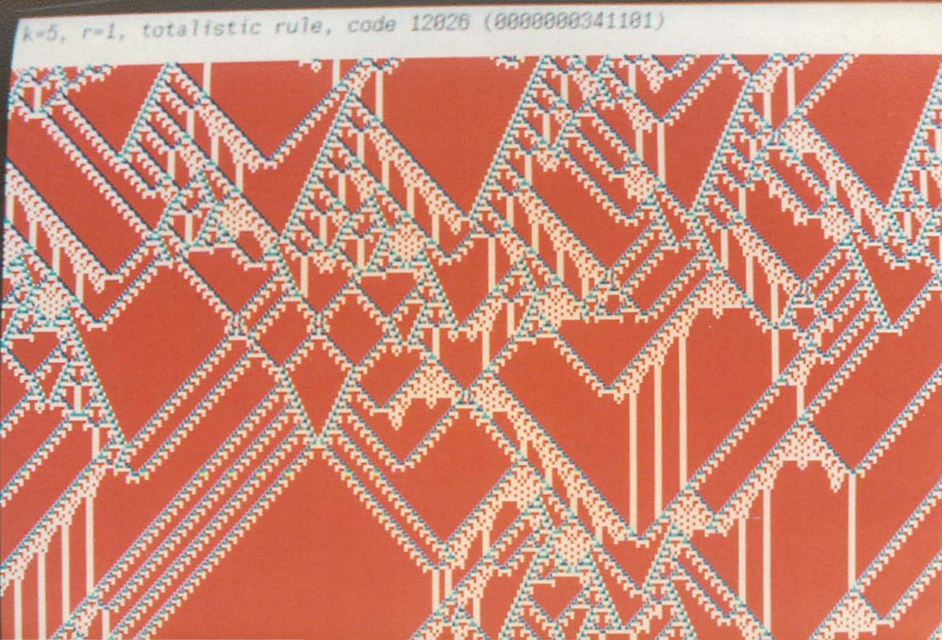
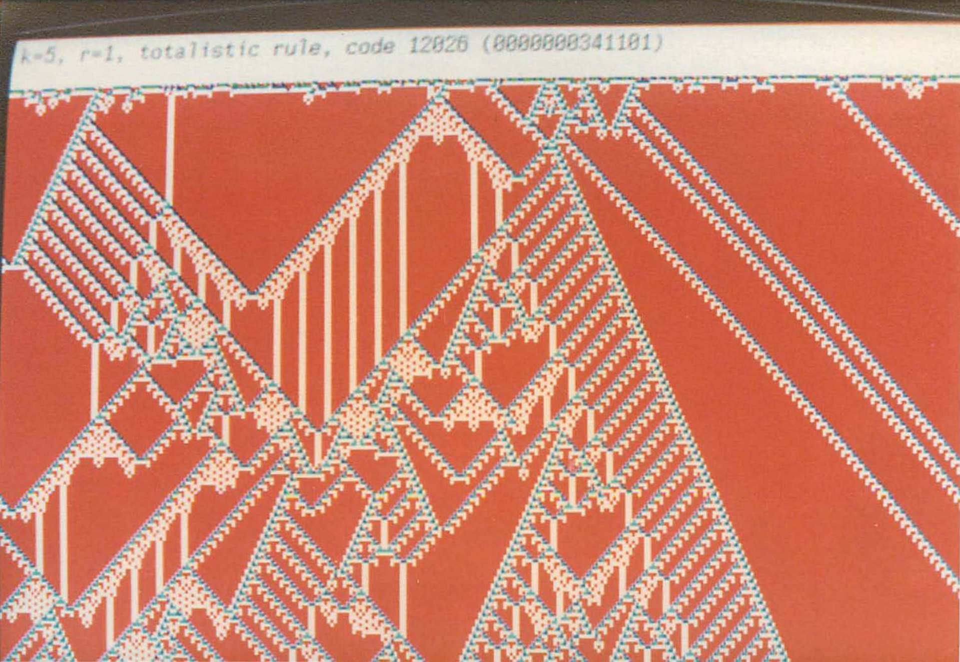
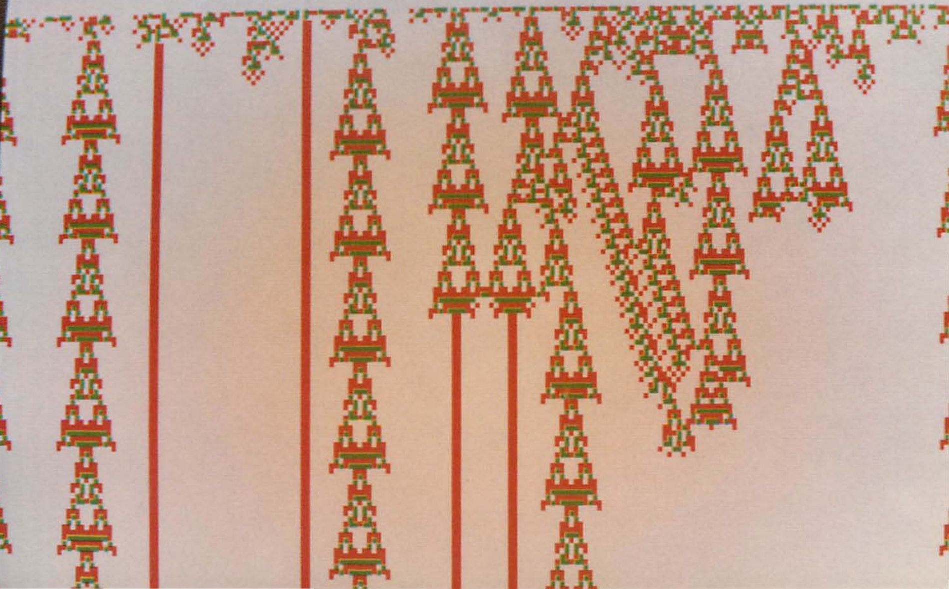

k=5, r=1, totalistic rule, code 12026 (0000000341101)

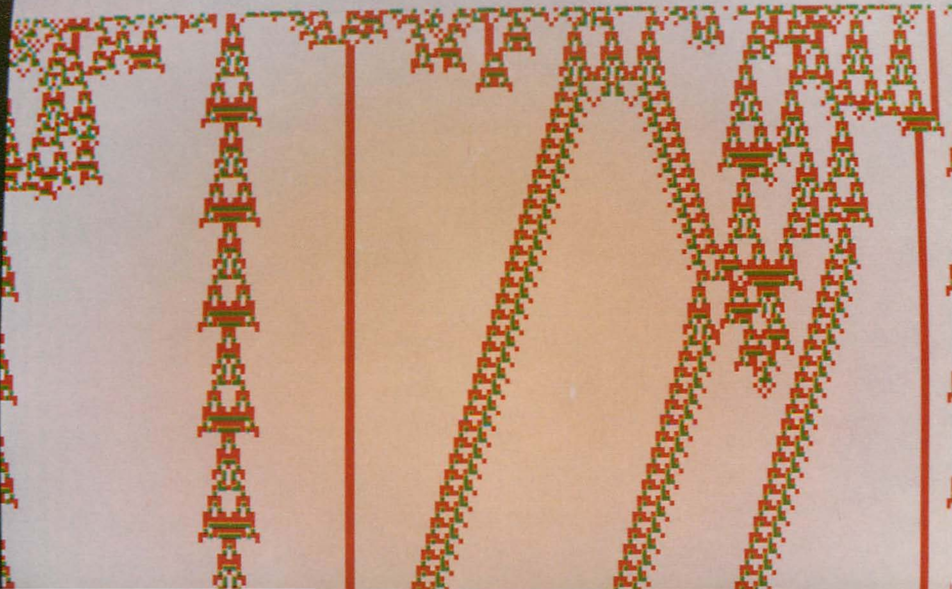


```
k=5, r=1, totalistic rule, code 12026 (0000000341101)
```

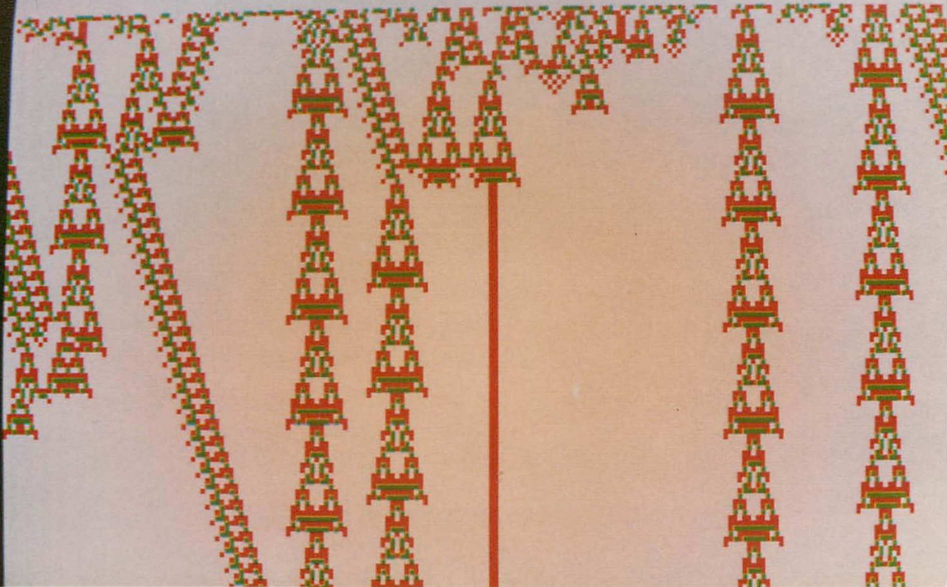
k=3, r=1, totalistic rule, code 792 (1002100)



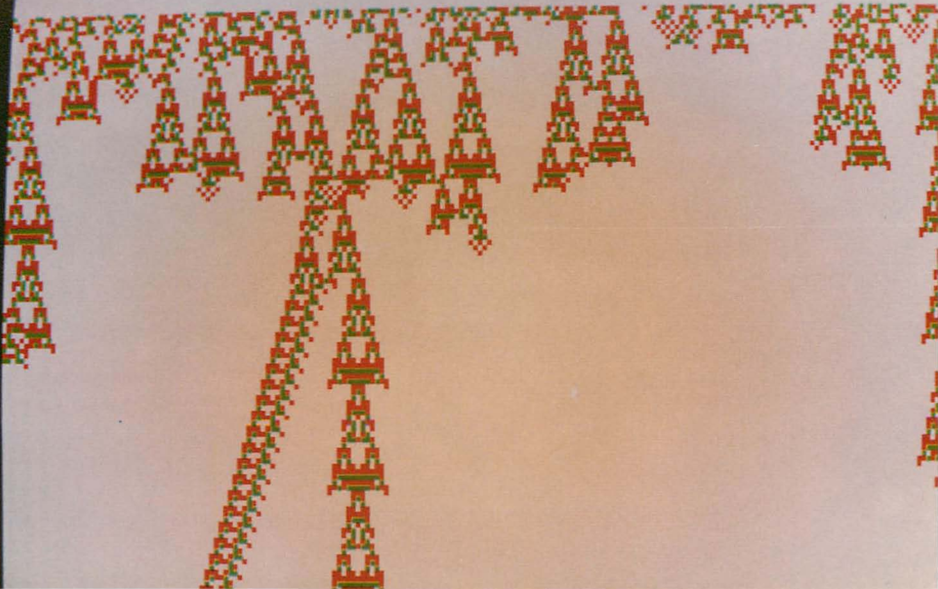
k=3, r=1, totalistic rule, code 792 (1002100)



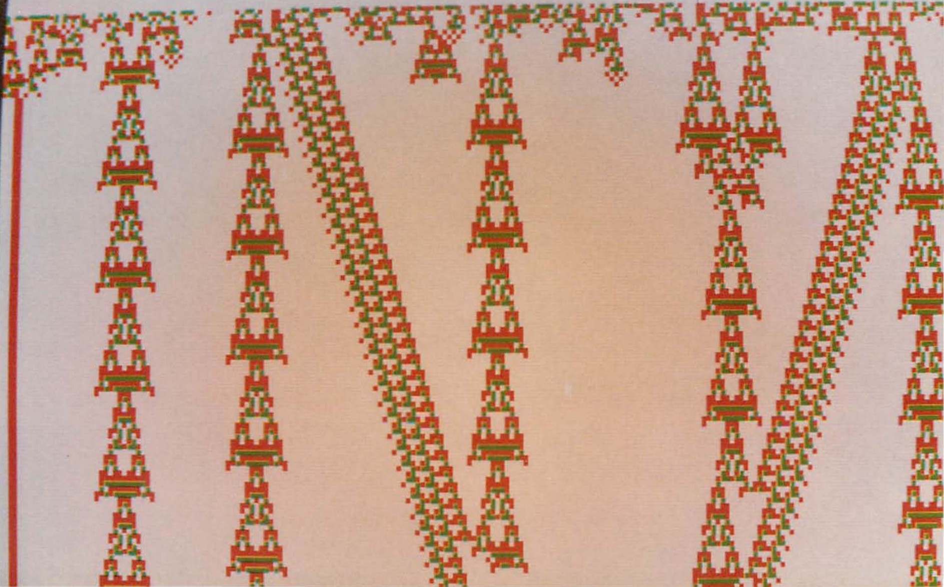
k=3, r=1, totalistic rule, code 792 (1002100)

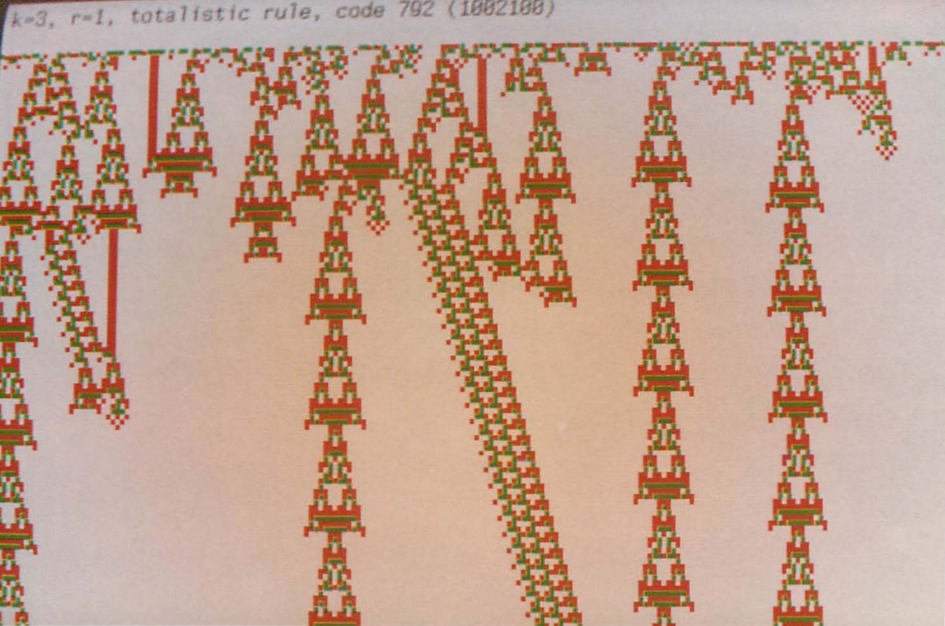


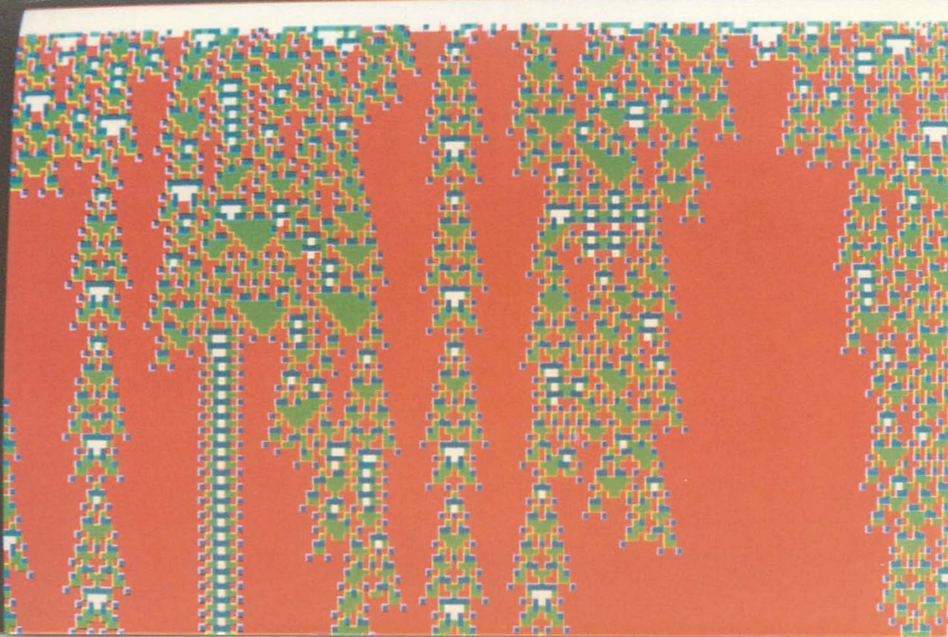
k=3, r=1, totalistic rule, code 792 (1002100)

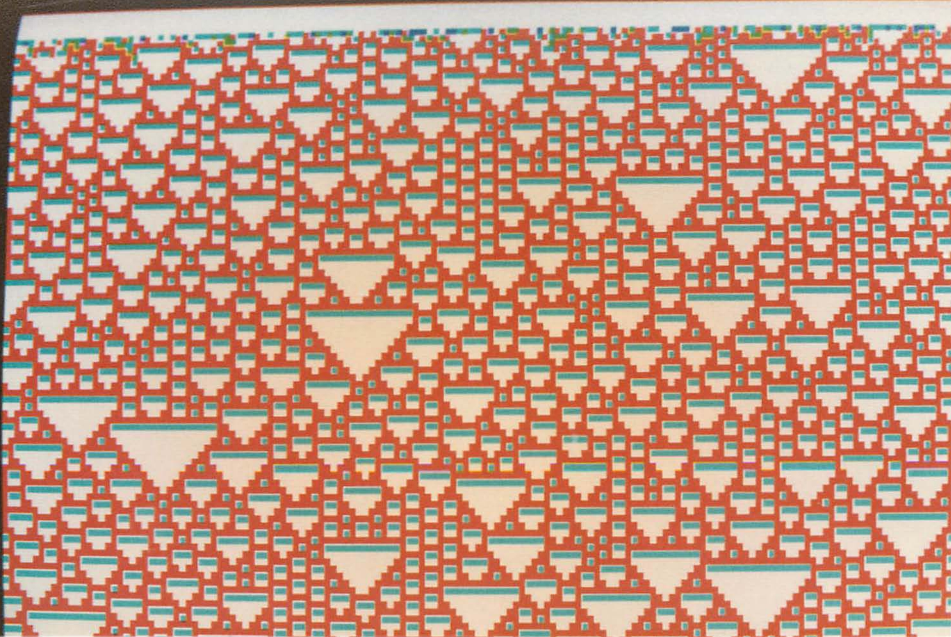


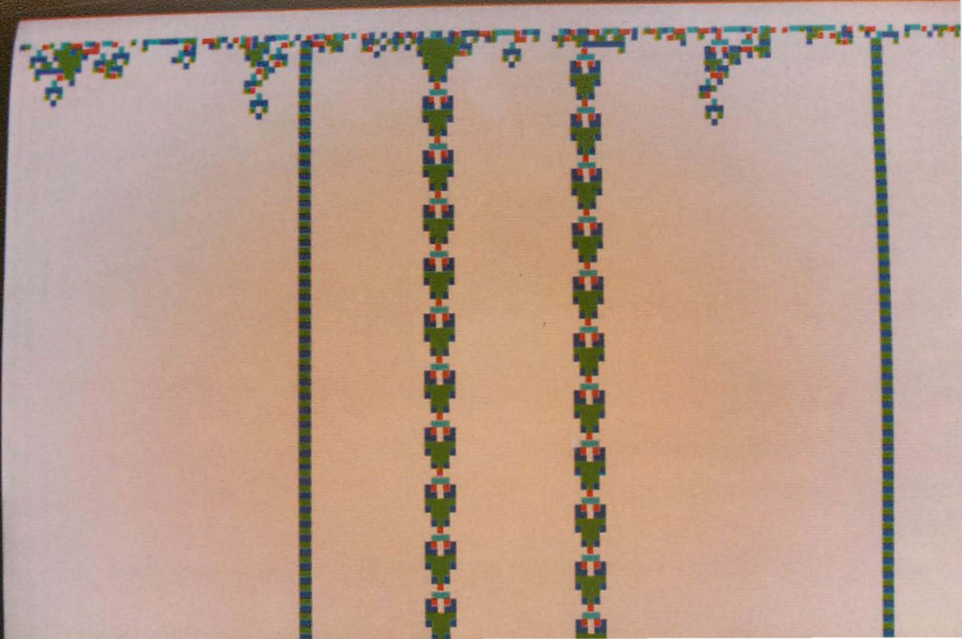
k=3, r=1, totalistic rule, code 792 (1002100)

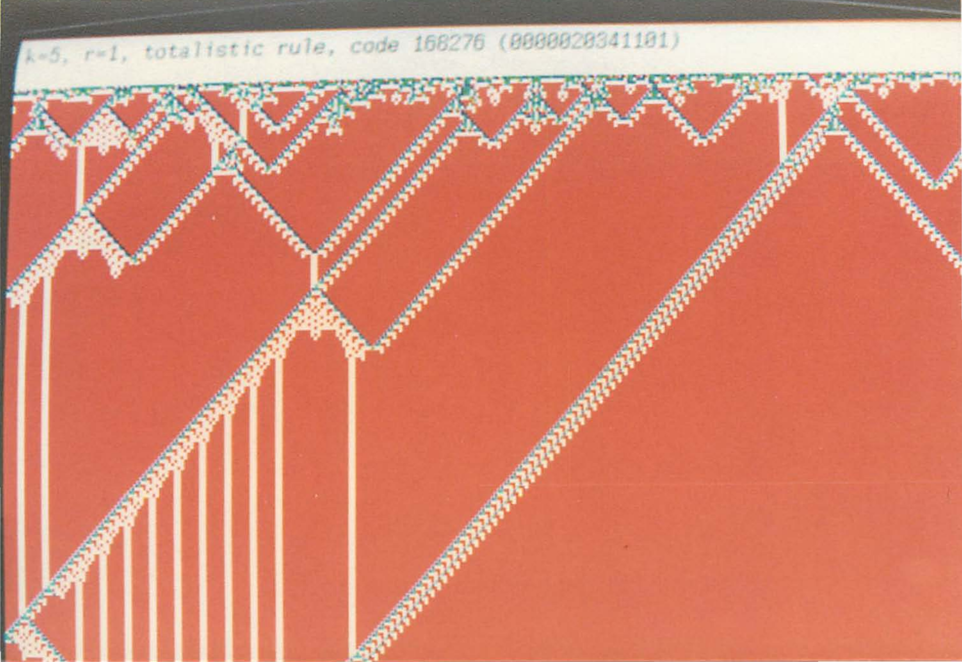


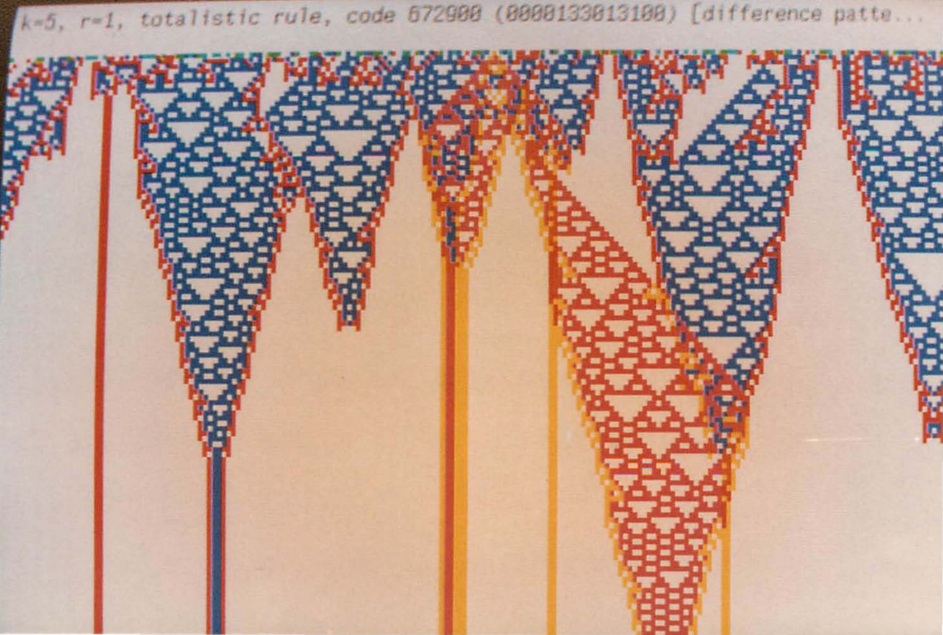

k=3, r=1, totalistic rule, code 792 (1002100)

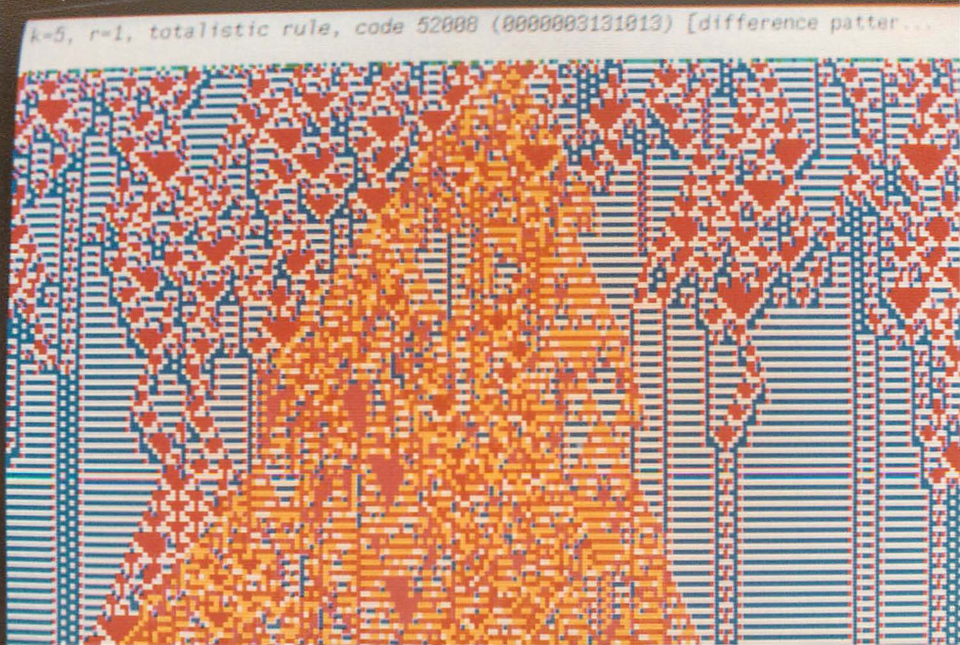




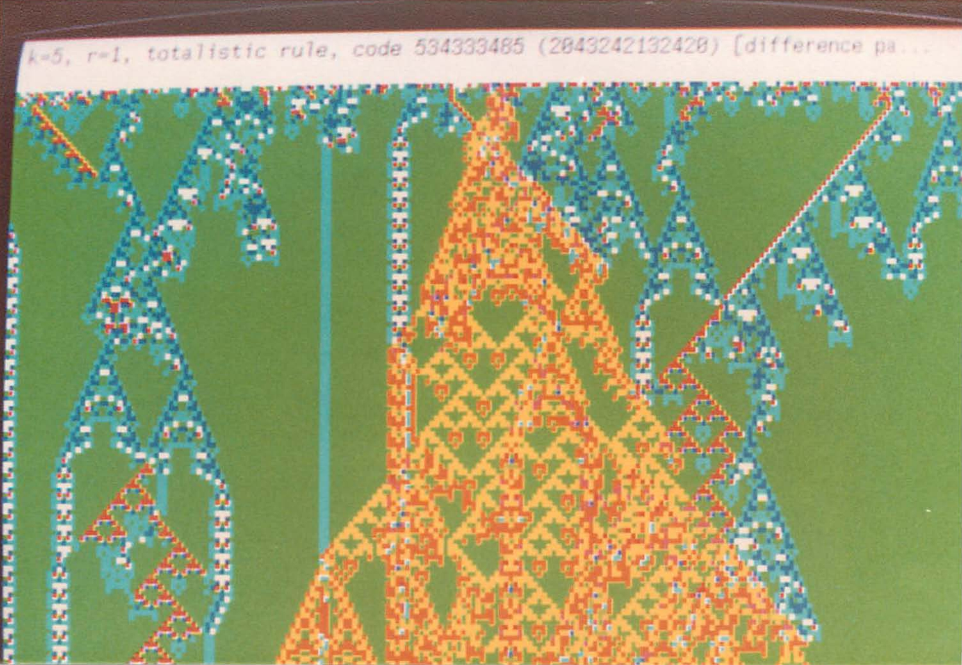


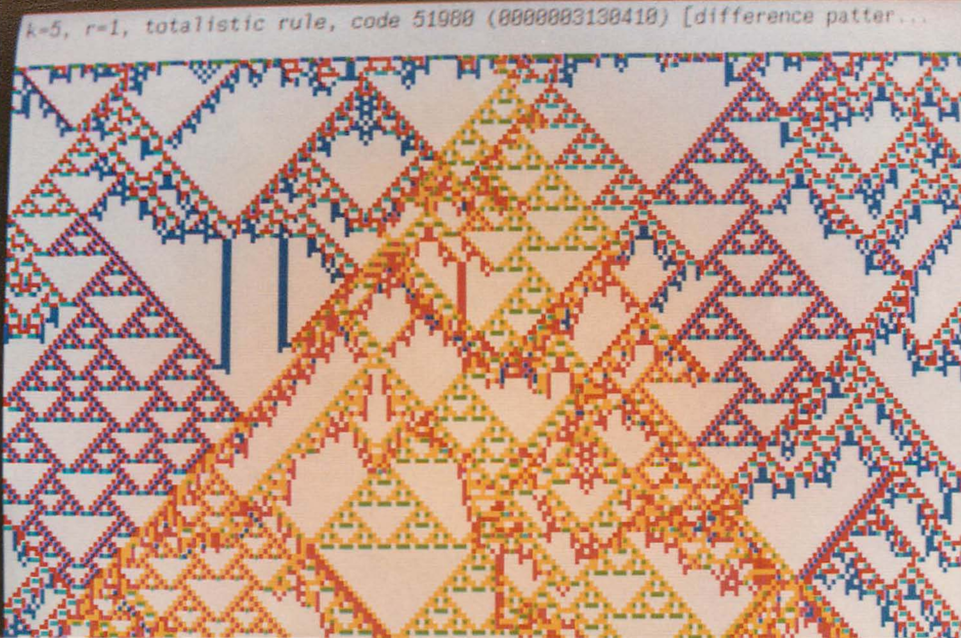

k=5, r=1, totalistic rule, code 168276 (0000020341101)


k=5, r=1, totalistic rule, code 672900 (0000133013100) [difference patte...

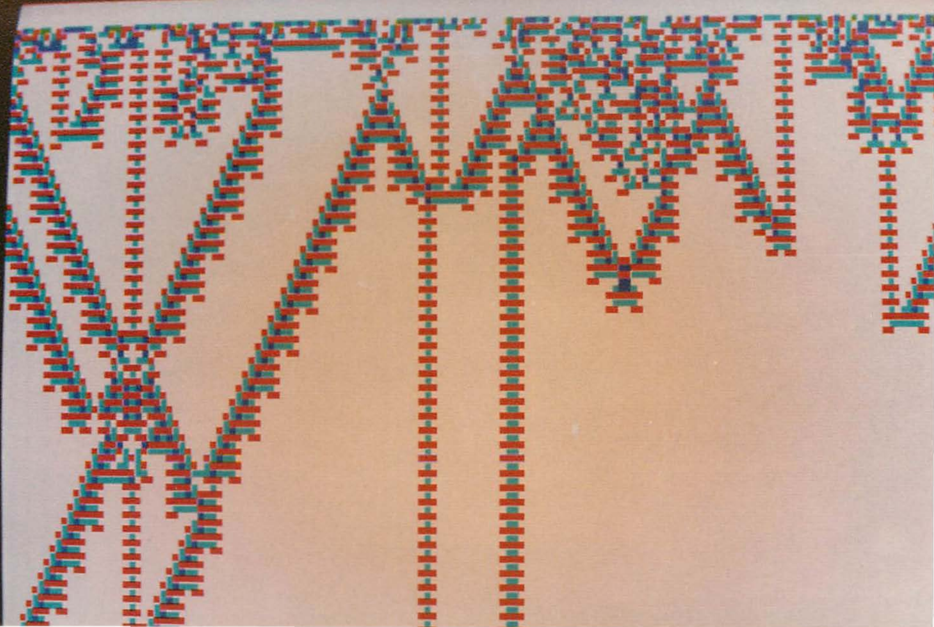


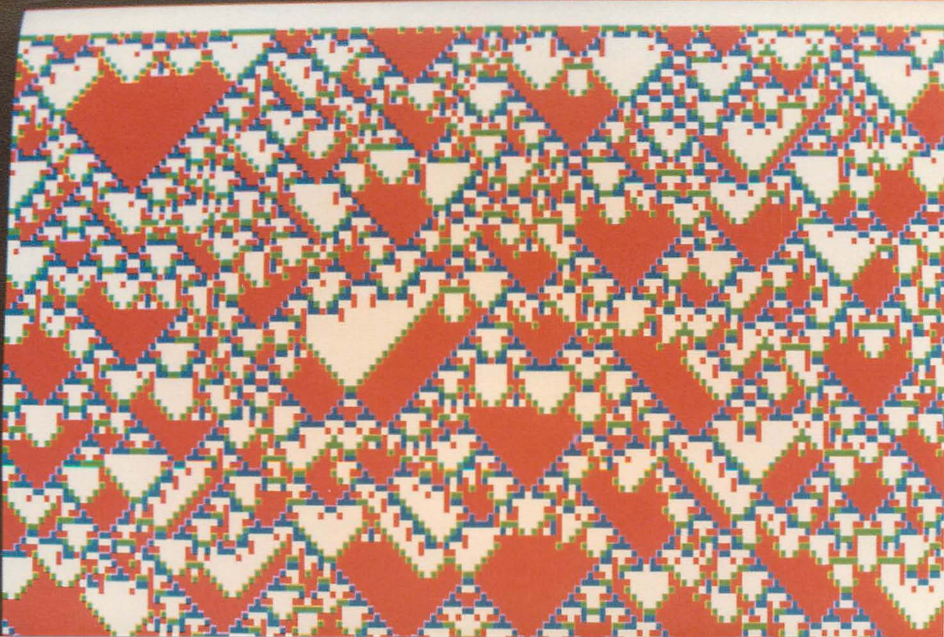
k=5, r=1, totalistic rule, code 52008 (00000003131013) [difference patter...

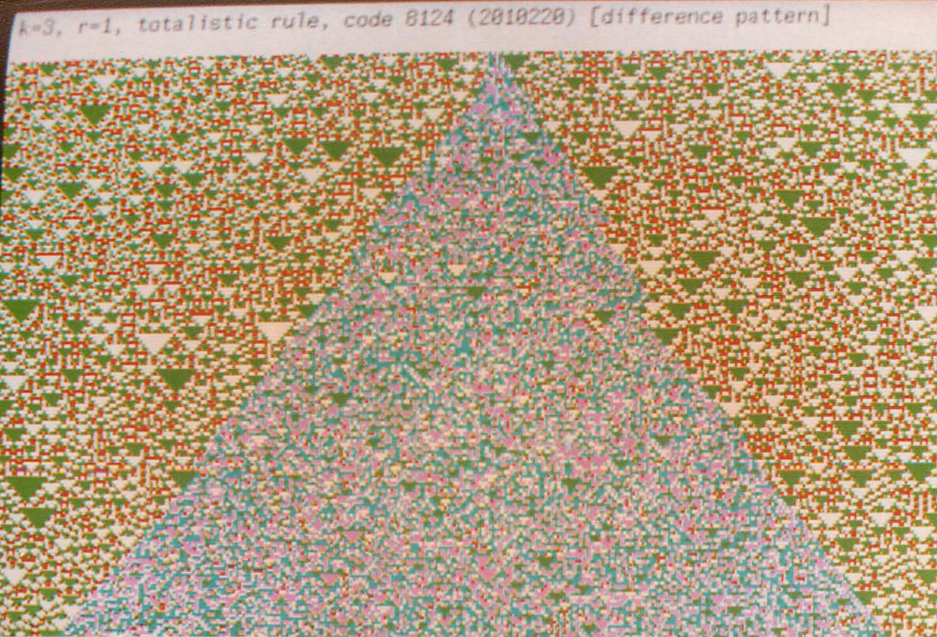

k=5, r=1, totalistic rule, code 534333485 (2043242132420) [difference pa...

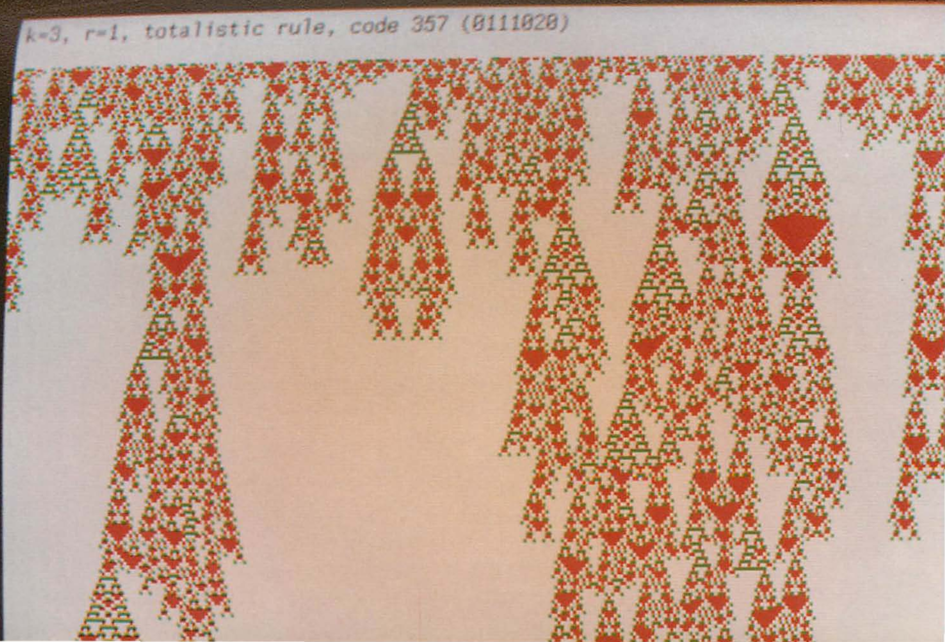


k=5, r=1, totalistic rule, code 51980 (0000003130410) [difference patter...






k=3, r=1, totalistic rule, code 8124 (2010220) [difference pattern]


k=3, r=1, totalistic rule, code 357 (0111020)

The Computer Museum

300 Congress Street
Boston, MA 02210
(617) 426-2800

September 21 1984

Gerard Vishniac
Lab for Computer Science
MIT
77 Massachusetts Ave
Cambridge
MA 02139

Dear Gerard

We spoke briefly on the 'phone about the cellular automata machine for the gallery here. I am now writing to ask you for the following photographs for our static displays:

1. Snowflake-type pattern

2. Wave-motion type pattern

3. Game of life example.

I have tried in vain to reach one of your group on the 'phone without success. We need the images by Wednesday of next week, September 26 at the absolute latest. Slides or prints will do, but the sooner you can get the material to us the better.

Please also let me know how you and Kent Multer are getting on. We have the IBM PC now. It has an IBM color graphics card in it and an IBM color monitor. I would like to use your machine in conjunction with 1-d examples being programmed for us by Stephen Wolfram, so we need to discuss the interface. Note that all interaction should be via numeric keys 0-9 and an enter key only.

Thank you for your participation. I look forward to hearing from you.

Yours sincerely

Oliver Strimpel

Oliver Strimpel
Curator

cc: Norman Margolis, Tomaso Toffoli, Multer

July 19 1984

Tomaso Toffoli
Lab for Computer Science
MIT
77 Mass Ave
Cambridge
MA 02139

Dear Tomaso

Further to our recent telephone conversation I enclose some information on The Computer Museum: The latest issue of *The Computer Museum Report*, the brochure prepared for our Capital Campaign, and an outline of the gallery *The Computer and the Image*.

We hope to integrate an IBM PC with your card into the Image gallery. Visitors would be invited to input either initial configuations or rules in some carefully controlled way so that they had a good chance of doing something satisfying in 5-10 minutes. We need to know the configuration required for the IBM PC as soon as possible.

I look forward to our meeting on August 1. I will call you that morning to confirm.

Yours sincerely

Dr Oliver Strimpel
Curator

enclosures

THE INSTITUTE FOR ADVANCED STUDY

PRINCETON, NEW JERSEY 08540

Telephone 609-734-8000

8080

SCHOOL OF NATURAL SCIENCES

September 25, 1984

Dr. Oliver Strimpel,
The Computer Museum,
300 Congress Street,
Boston, MA 02210.

Dear Oliver,

Here are the text and slides for the cellular automaton exhibit. I have enclosed eight one-dimensional slides. It might be nice to use all eight, but perhaps it would be best to have them printed and see which six look best together. I have also enclosed a two-dimensional snowflake pattern. If necessary, we could generate other two-dimensional patterns (they just run rather slowly on our computer).

On the one-dimensional slides: each slide should be cropped to remove the caption. The ones with simple initial states should be arranged so that the bottom of the pattern coincides with the bottom of the picture; the top of the pattern should however be somewhat below the top of the picture, I think. For the patterns with random initial states, the bottom of the picture should again coincide with the bottom of the pattern. In so far as the top of the pattern is not straight (due to the curvature of the TV screen), it could be cropped off.

Please note that the one-dimensional patterns are copyrighted by me; the snowflake pattern is copyrighted by Norman H. Packard.

We are trying to work on the software this week. I am not sure if we will get it done by the end of this week, but I expect to have it by the beginning of next week.

Regards,

Stephen

Stephen Wolfram.

Cellular Automata

Complex patterns found in nature are often built up from simple components. Cellular automata are mathematical models in which complex patterns are formed by the repeated action of simple rules. Computer simulation and computer graphics are essential tools in their investigation. ~~This exhibit shows some examples of the patterns they produce.~~

(8 1D pictures; © 1984 Stephen Wolfram) One-dimensional

One-dimensional cellular automata ~~(illustrated above)~~ consist of a line of cells. The colour of each cell is chosen ~~from a few possibilities~~ according to a rule that depends on the colours of the cells immediately around it on the line above. Even though the rules are simple, the patterns produced by applying them over and over again can be highly complex. The top row of pictures show the results of growth ~~from a seed consisting of~~ a single red cell. The bottom row shows evolution from a random initial state. The rules illustrated gives examples of several of the classes of behaviour found. But there are an immense number of possible cellular automaton rules. With the computer demonstration you can select and study your own rule.

few

starting from

(4 2D pictures)

Two-dimensional cellular automata consist of a grid of cells. The pictures above are still frames from a computer simulation of their evolution. The top left-hand picture is an example of the notorious "Game of Life" - a two-dimensional cellular automaton with very varied behaviour studied by recreational mathematicians for over a decade. The other pictures show cellular automaton models for various natural phenomena: snowflakes, ripples on a pond, and water flowing past a cylinder.

In the computer demonstration of one-dimensional cellular automata, five colours are used, corresponding to values zero through four for each cell (0:black, 1:red, 2:green; 3:blue, 4:~~yellow~~ ???). The rules take the value of a cell to be determined by the sum of its own previous value, and the previous values of its immediate neighbours to the left and right: $a_i' = f(a_{i-1} + a_i + a_{i+1})$. The function $f(n)$ specified by a code number. The code number is initially computed in base five. Starting from the right-hand end, the first digit gives $f(0)$, the next $f(1)$, and so on; the left-hand digit is $f(12)$. The code numbers are entered in decimal, then converted to base five.

?

For further information, see:

S. Wolfram, "Computer software in science and mathematics", *Scientific American*, September 1984, p. 188;

S. Wolfram, "Cellular automata as models of complexity", *Nature*, XXXXXXXXXXXXXX.

to be placed on screen.

? include rule numbers for pictures in caption.

MATHEMATICAL GAMES

On cellular automata, self-reproduction, the Garden of Eden and the game "life"

by Martin Gardner

John Horton Conway's game "life," last October's topic, stirred such interest among computer scientists that this month's department will again be devoted to the game. Before reporting as many new discoveries as possible I should like to discuss some highlights in the history of "cellular automata theory," the field in which games similar to Conway's are being investigated.

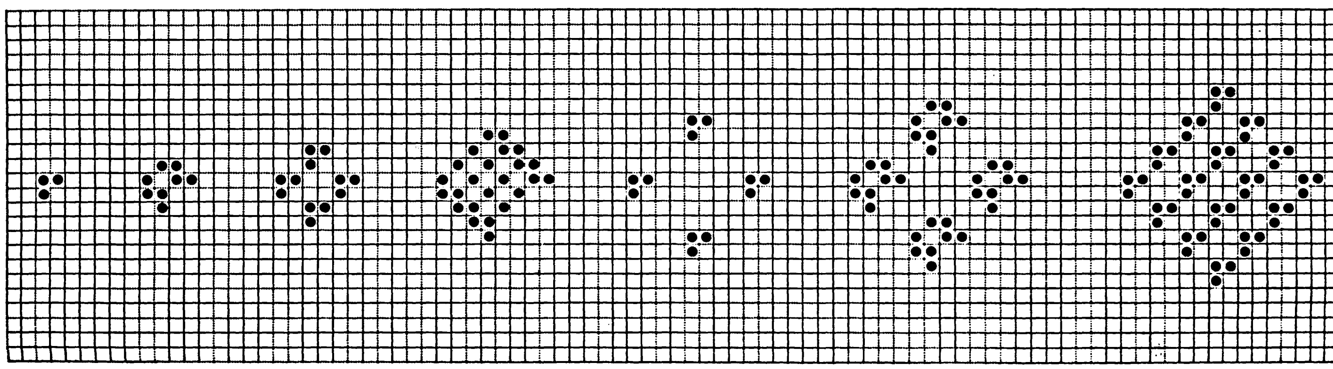
It all began about 1950, when John von Neumann set himself the task of proving the possibility of self-duplicating automata. Such a machine, given proper instructions, would build an exact duplicate of itself. Each of the two machines would then build another, the four would become eight, and so on. (This proliferation of self-replicating automata is the basis of Lord Dunsany's amusing 1951 novel *The Last Revolution.*) Von Neumann first proved his case with "kinematic" models of a machine that could roam through a warehouse of parts, select needed components and put together a copy of itself. Later, adopting an inspired suggestion by his friend Stanislaw M. Ulam, he showed the possibility of such machines in a more elegant and abstract way.

Von Neumann's new proof used what is now called a "uniform cellular space" equivalent to an infinite checkerboard. Each cell can have any finite number of "states," including a "quiescent" (or empty) state, and a finite set of "neighbor" cells that can influence its state. The pattern of states changes in discrete time steps according to a set of "transition rules" that apply simultaneously to every cell. The cells symbolize the basic parts of a finite-state automaton and a configuration of live cells is an idealized model of such a machine. Conway's game is based on just such a space. His neighborhood consists of the eight cells surrounding a cell; each cell has two states (empty or filled), and his transition rules are the birth, death and survival rules I explained in October. Von Neumann, applying transition rules to a space in which each cell has 29 states and four orthogonally adjacent neighbors, proved the existence of a configuration of about 200,000 cells that would self-reproduce.

The reason for such an enormous configuration is that, for von Neumann's proof to apply to actual automata, it was necessary that his cellular space be capable of simulating a Turing machine: an idealized automaton, named for its inventor, the British mathematician A. M. Turing, capable of performing any desired calculation. By embedding this universal computer in his configuration, von Neumann was able to produce a universal constructor. Because it could in principle construct any desired configuration by stretching "arms" into an empty region of the cellular space, it would self-replicate when given a blueprint of itself. Since von Neumann's death in 1957 his existence proof (the actual configuration is too vast to construct and manipulate) has been greatly simplified. The latest and best reduction, by Edwin Roger Banks, a mechanical engineering graduate student at the Massachusetts Institute of Technology, does the job with cells of only four states.

Self-replication in a trivial sense—without using configurations that contain Turing machines—is easy to achieve. A delightfully simple example, discovered by Edward Fredkin of M.I.T. about 10 years ago, uses two-state cells, the von Neumann neighborhood of four orthogonally adjacent cells and the following parity rule: Each cell with an even number of live neighbors (0, 2, 4) at time t becomes or remains empty at time $t + 1$, and each cell with an odd number of neighbors (1, 3) at time t becomes or remains live at time $t + 1$. It is not hard to show that after 2^n moves (n varying with different patterns) any initial pattern of live cells will reproduce itself four times—above, below, left and right of an empty space that it formerly occupied. The four replicas will be displaced 2^n cells from the vanished original. The new pattern will, of course, replicate again after another 2^n steps, so that the duplicates keep quadrupling in the endless series 1, 4, 16, 64, ... The illustration below shows two quadruplings of a right tromino. Terry Winograd, in a 1967 term paper written when he was an M.I.T. student, generalized Fredkin's rule to other neighborhoods, any number of dimensions and cells with any prime number of states.

Ulam investigated a variety of cellular automata games, experimenting with different neighborhoods, numbers of states and transition rules. In a 1967 paper "On Recursively Defined Geomet-


The replication of a tromino

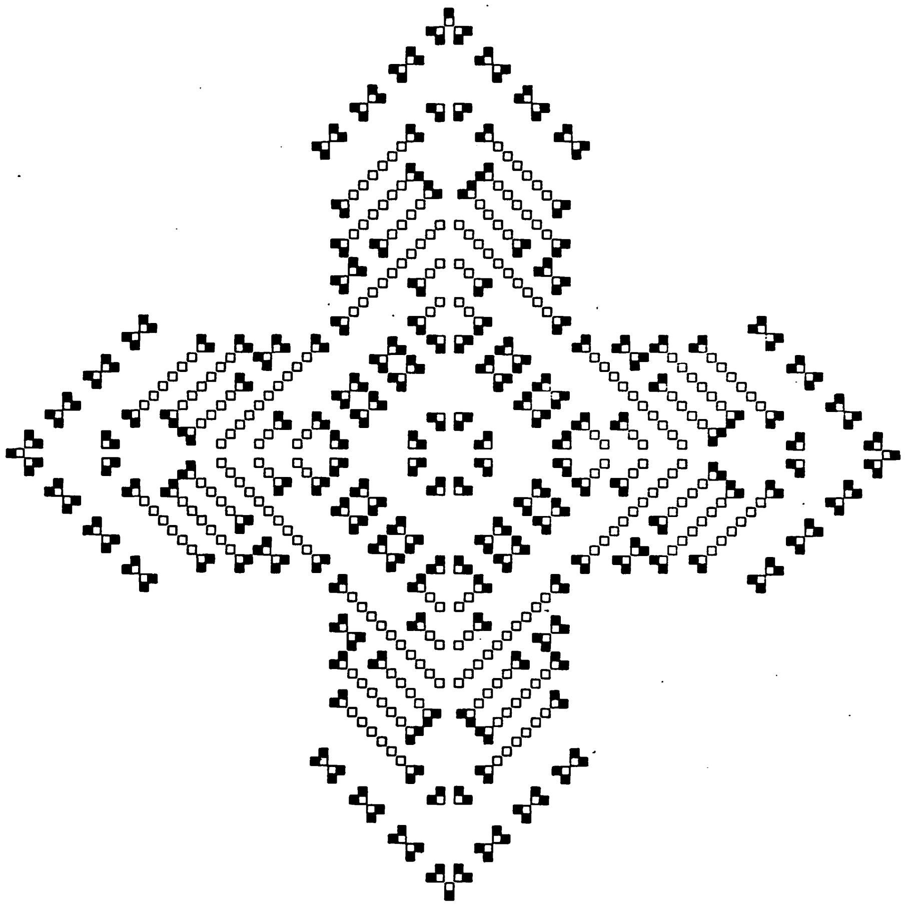
rical Objects and Patterns of Growth," written with Robert G. Schrandt, Ulam described a number of different games. The upper illustration on this page shows generation 45 of a history that began with one counter on the central cell. As in Conway's game, the cells are two-state, but the neighborhood is that of von Neumann (four adjacent orthogonal cells). Births occur on cells that have one and only one neighbor, and all live cells of generation n vanish when generation $n + 2$ is born. In other words, only the last two generations survive at any step. In the illustration the 444 new births are shown as black cells. The 404 white cells of the preceding generation will all disappear on the next move. Note the characteristic subpattern, which Ulam calls a "dog bone." Ulam experimented with games in which two configurations were allowed to grow until they collided. In the ensuing "battle" one side would sometimes wipe out the other; sometimes both armies would be annihilated. Ulam also explored games on three-dimensional cubical tessellations. His major papers on cellular automata are in *Essays on Cellular Automata* (University of Illinois Press, 1970), edited by Arthur W. Burks.

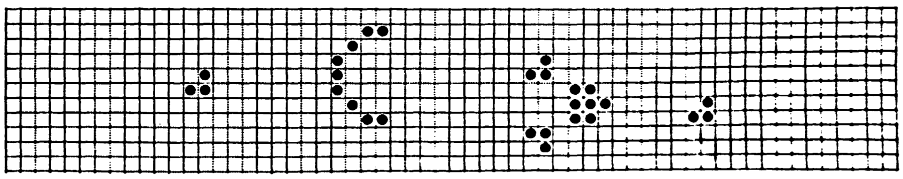
Similar games can be devised for triangular and hexagonal tessellations but, although they *look* different, they are not essentially so. All can be translated into equivalent games on a square tessellation by a suitable definition of "neighborhood." A neighborhood need not be made up of touching cells. In chess, for instance, a knight's neighborhood consists of the squares to which it can leap and squares on which there are pieces that can attack it. As Burks has pointed out, games such as chess, checkers and go can be regarded as cellular automata games in which there are complicated neighborhoods and transition rules and in which players choose among alternative next states in an attempt to be first to reach a certain final state that wins.

Among the notable contributions of Edward F. Moore to cellular automata theory the best-known is a technique for proving the existence of what John W. Tukey named "Garden of Eden" patterns. These are configurations that cannot arise in a game because no preceding generation can form them. They appear only if given in the initial (zero) generation. Because such a configuration has no predecessor, it cannot be self-reproducing. I shall not describe Moore's ingenious technique because he explained it informally in an article in this magazine [see "Mathematics in the Biological Sciences," by Edward F. Moore; September, 1964] and more formally in a paper that is included in Burks's anthology.

Alvy Ray Smith III, a cellular automata expert at New York University's School of Engineering and Science, found a simple application of Moore's technique to Conway's game. Consider two five-by-five squares, one with all cells empty, the other with one counter in the center. Because, in one move, the central nine cells of both squares are certain to become identical (in this case all cells empty) they are said to be "mutually erasable." It follows from Moore's theorem that a Garden of Eden configuration must exist in Conway's game. Unfortunately the proof does not tell how to find such a pattern and so far none is known. It may be simple or it may be enormously complex. Using one of Moore's formulas, Smith has been able to calculate that such a pattern exists within a square of 10 billion cells on a side, which does not help much in finding one.

Smith has been working on cellular automata that simulate pattern-recognition machines. Although this is now only of theoretical interest, the time may come when robots will need "retinas" for recognizing patterns. The speeds of scanning devices are slow compared with the speeds obtainable by the "parallel computation" of animal retinas, which simultaneously transmit thousands of messages to the brain. Parallel computation is the only way new computers can increase significantly in speed because without it they are limited by the speed of light through miniaturized circuitry [see "The Fastest Computer," by D. L.


Generation 45 in a cellular game devised by Stanislaw M. Ulam


A configuration that grows into a glider gun

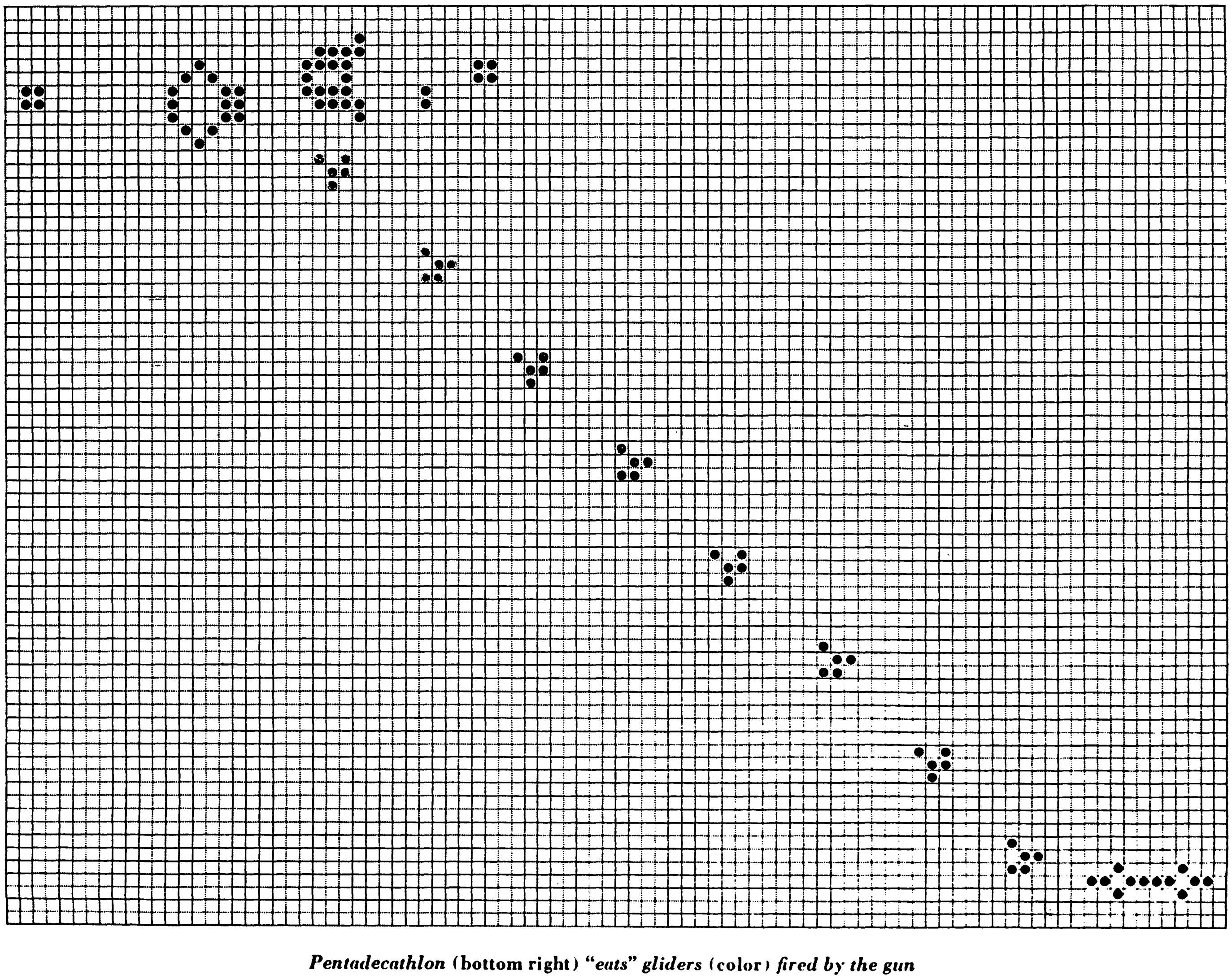
Slotnick, page 76]. The cover of this issue of SCIENTIFIC AMERICAN shows a simple procedure, devised by Smith, by which a finite one-dimensional cellular space employs parallel computation for recognizing palindromic symmetry. Each cell has many possible states, the number depending on the number of different symbols in the palindrome, and a cell's neighborhood is the two cells on each side.

On the cover, which Smith designed, he symbolizes the palindrome TOO HOT TO HOOT with four states of cells in the top row. *T*, *O* and *H* are represented by blue, red and yellow respectively, and black marks the palindrome's two ends. The white cells in the other rows are in the quiescent state. The horizontal rows below the top row are successive generations of the top configuration when certain transition rules are followed in discrete time steps. In other words, the picture is a space-time diagram of a single row, each successive row indicating the next generation.

In the first transition each color travels one cell to the left and one cell to the right, except for the end colors, which are blocked by black; black moves inward at each step. Each cell on which two colors land acquires a new state, symbolized by a cell divided into four triangles. The left triangle has the color that was previously on the left, the right triangle has the color previously on the right. The result of this first move is shown in the second row. When an adjacent pair of cells forms a tilted square in the center that is a solid color, it indicates a "collision" of like colors and is symbolized by black dots in the two white triangles of the left cell. Dots remain in that cell for all subsequent generations unless a collision of unlike colors occurs to the immediate right of the dotted cell, in which case the dots are erased. When collisions of *unlike* colors occur, the left cell of the pair remains undotted for all subsequent generations even though like colors may later collide on its right.

At each move the colors continue to travel one cell left or right (the direction in which the colored triangles point) and all rules apply. If the palindrome has n letters, with n even as in this example (the scheme is modified slightly if n is odd), it is easy to see that after $n/2$ moves only two adjacent nonquiescent cells remain. If the left cell of this pair is dotted, the automaton has recognized the initial row as being palindromic. Down the diagram's center you see the colliding pairs of like colors in the same order as they appear on the palindrome from the center to each end. As soon as recognition occurs the left cell of the last pair is erased and the right cell is altered to an "accept" state, here symbolized by nested squares. An undotted left cell would signal a nonpalindrome, in which case the left cell would become blank and the right cell would go into a "reject" state.

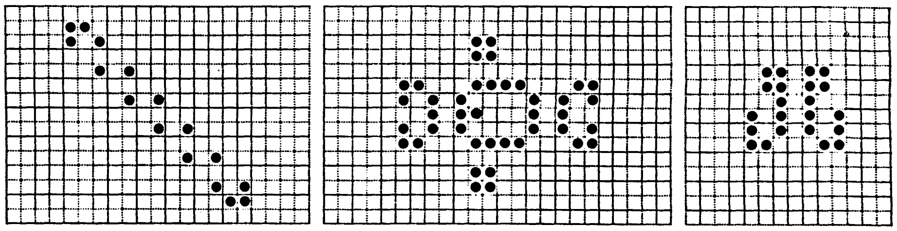
A Turing machine, which computes serially, requires in general n^2 steps to recognize a palindrome of length n. Al-

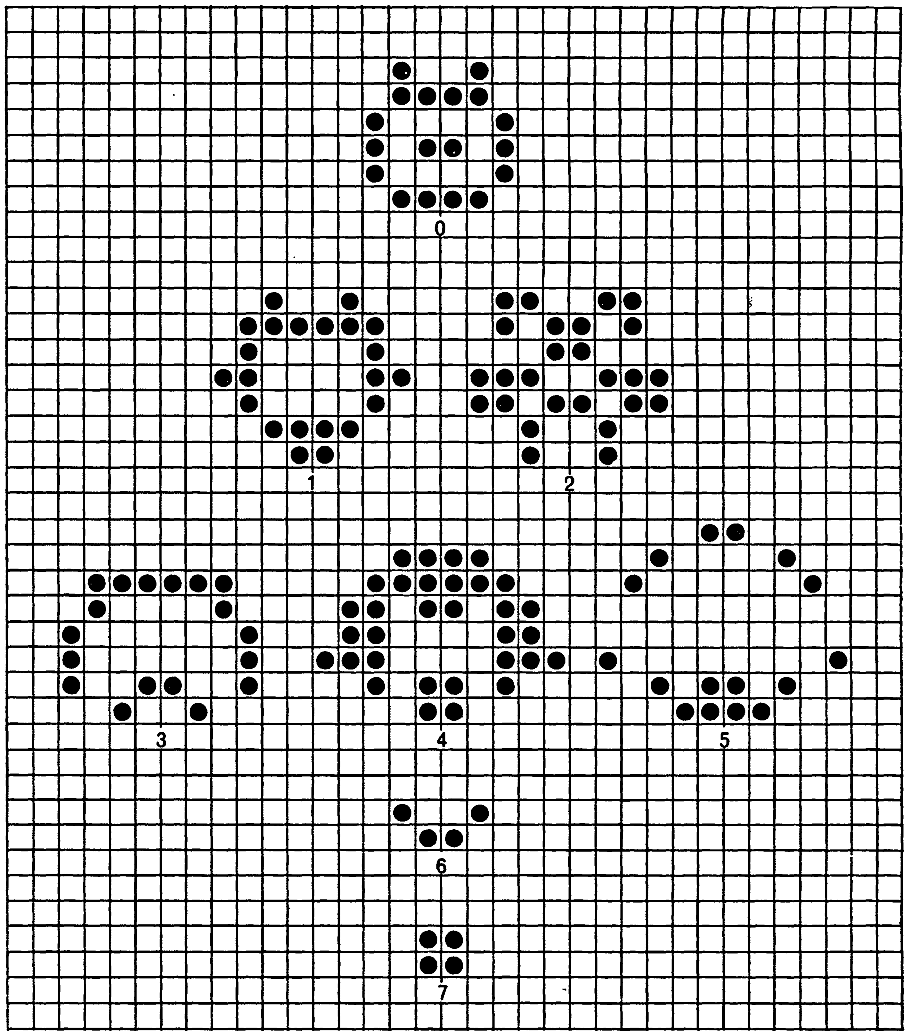


Pentadecathlon (bottom right) *"eats" gliders* (color) *fired by the gun* though recognition occurs here at step $n/2$, the accept state is shown moving in subsequent generations to the right to symbolize the cell-by-cell transmission of the acceptance to an output boundary of the cellular space. Of course it is easy to construct more efficient palindrome-recognizing devices with actual electronic hardware, but the point here is to do it with a highly abstract, one-dimensional cellular space in which information can pass only from a cell to adjacent cells and not even the center of the initial series of symbols is known at the outset. As Smith puts it anthropomorphically, after the first step each of the three dotted cells thinks it is at the center of a palindrome. The dotted cells at each end are disillusioned on the next move because of the collision of unlike colors at their right. Not until generation $n/2$ does the dotted cell at the center know it actually *is* at the center.

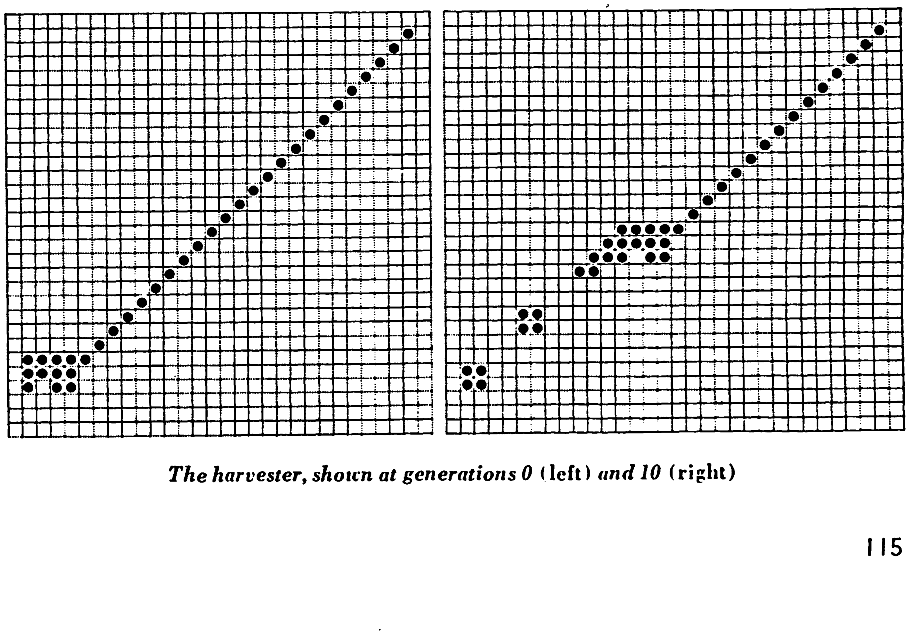
Now for some startling new results concerning Conway's game. Conway was fully aware of earlier games and it was with them in mind that he selected his recursive rules with great care to avoid two extremes: too many patterns that grow quickly without limit and too many that fade quickly. By striking a delicate balance he designed a game of surprising unpredictability and one that produced such remarkable figures as oscillators and moving spaceships. He conjectured that no finite population could grow (in number of members) without limit, and he offered $50 for the first proof or disproof. The prize was won in November by a group in the Artificial Intelligence Project at M.I.T. consisting of (in alphabetical order) Robert April, Michael Beeler, R. William Gosper, Jr., Richard Howell, Rich Schroeppel and Michael Speciner. Using a program devised by Speciner for displaying moves on an oscilloscope, Gosper made a truly astounding discovery: he found a glider gun! The configuration in the lower illustration on page 113 grows into such a gun, firing its first glider on move 40. The gun is an oscillator of period 30 that ejects a new glider every 30 moves. Since each glider adds five more counters to the field, the population obviously grows without limit.

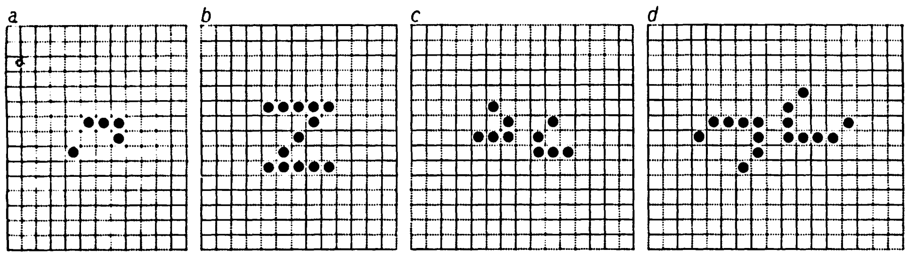
The glider gun led the M.I.T. group to many other amazing discoveries. A series of printouts (supplied by Robert T. Wainwright of Yorktown Heights, N.Y.) shows how 13 gliders crash to form a glider gun [*see illustration on page 117*]. The last five printouts show the gun in full action. The group also found a way to position a pentadecathlon [*see*


Barber pole (left), *Hertz oscillator* (middle) *and tumbler* (right)



The Cheshire cat (0) *fades to a grin* (6) *and disappears, leaving a paw print* (7)


The harvester, shown at generations 0 (left) *and 10* (right)



Two spawners of gliders and two collision courses

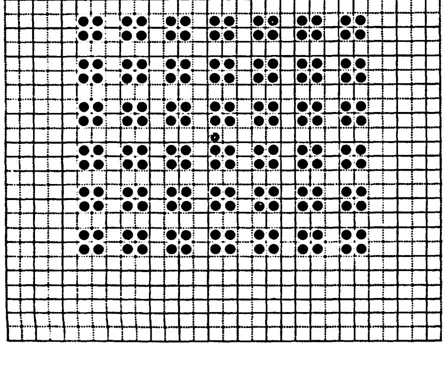
illustration on page 114], an oscillator of period 15, so that it "eats" every glider that strikes it. A pentadecathlon can also reflect a glider 180 degrees, making it possible for two pentadecathlons to shuttle a glider back and forth forever. Streams of intersecting gliders produce fantastic results. Strange patterns can be created that in turn emit gliders. Sometimes collision configurations grow until they ingest all guns. In other cases the collision mass destroys one or more guns by shooting back. The group's latest burst of virtuosity is a way of placing guns so that the intersecting streams of gliders build a factory that assembles and fires a lightweight spaceship every 300 moves.

The existence of glider guns raises the exciting possibility that Conway's game will allow the simulation of a Turing machine, a universal calculator capable in principle of doing anything the most powerful computer can do. The trick would be to use gliders as unit pulses for storing and transmitting information and performing the required logic operations that are handled in actual computers by their circuitry. If Conway's game allows a universal calculator, the next question will be whether it allows a universal constructor, from which nontrivial self-replication would follow. So far this has not been achieved with a two-state space and Conway's neighborhood, although it has been proved impossible with two states and the von Neumann neighborhood.

The M.I.T. group found many new oscillators [*see top illustration on preceding page*]. One of them, the barber pole, can be stretched to any length and is a flip-flop, with each state a mirror image of the other. Another, which they rediscovered, is a pattern Conway's group had found earlier and called a Hertz oscillator. Every four moves the colored "bit" switches from one side of the central frame to the other, making it an oscillator of period 8. The tumbler, which was found by George D. Collins, Jr., of McLean, Va., turns upside down every seven moves.

The Cheshire cat [*see middle illustration on preceding page*] was discovered by C. R. Tompkins of Corona, Calif. On the sixth move the face vanishes, leaving only a grin; the grin fades on the next move and only a permanent paw print (block) remains. The harvester was constructed by David W. Poyner of Basildon in England. It plows up an infinite diagonal at the speed of light, oscillating with period 4 and ejecting stable packages along the way [*see bottom illustration on preceding page*]. "Unfortunately," writes Poyner, "I have been unable to develop a propagator that will sow as fast as the harvester will reap."

Wainwright has made a number of intriguing investigations. He filled a 120-by-120 square field with 4,800 randomly placed bits (a density of one-third) and tracked their history for 450 generations, by which time the density of this primordial soup, as Wainwright calls it, had thinned steadily to one-sixth. Whether it would eventually vanish or, as Wainwright says, percolate at a constant minimum density is anybody's guess. At any rate, during the 450 generations 42 short-lived gliders were formed. Wainwright found 14 different patterns that became glider states on the next move. The pattern that produced the greatest number of gliders (14 in all) is shown [*"a" in illustration above*]. A Z-pattern found by Collins and by Jeffrey Lund of Pewaukee, Wis., after 12 moves becomes two gliders that sail off in opposite directions [*b*]. Wainwright and others set two gliders on a collision course that causes all bits to vanish on the fourth move [*c*]. Wallace W. Wagner of Anaheim, Calif., found a collision course for two lightweight spaceships that also ends (on the seventh move) in total blankness [*d*].



Pattern doomed by a virus (color)

Wainwright has experimented with various infinite fields of regular stable patterns, which he calls agars—rich culture mediums. When, for instance, a single "virus," or bit, is placed in the agar of blocks shown in the illustration at bottom left so that it touches the corners of four blocks, the agar eliminates the virus and repairs itself in two moves. If, however, the alien bit is positioned as shown (or at any of the seven other symmetrically equivalent spots), it initiates an inexorable disintegration of the pattern. The portion eaten away contains active debris that has overall bilateral symmetry along one axis and a roughly oval border that expands, probably forever, in the four compass directions at the speed of light.

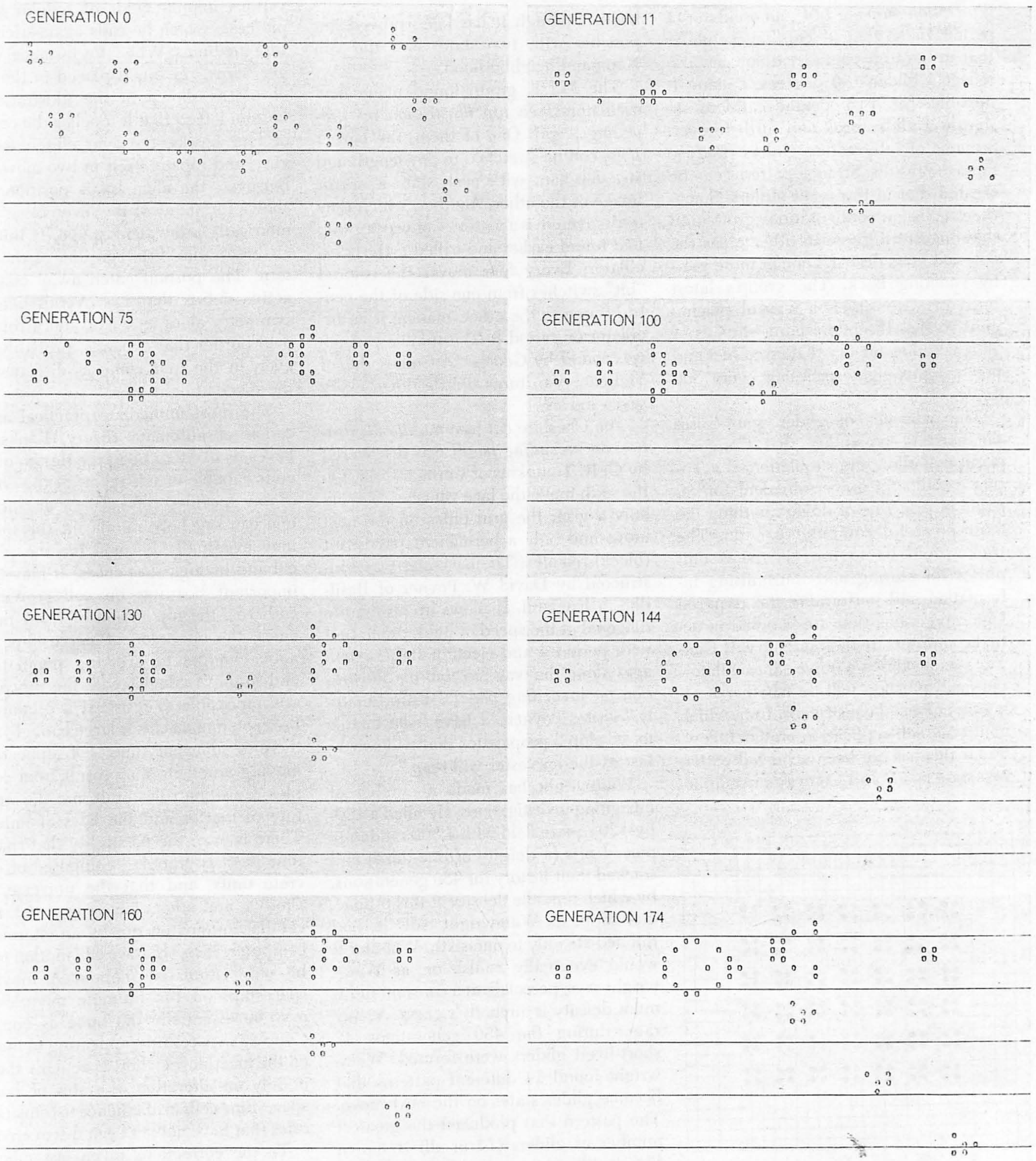
The most immediate practical application of automata theory, Banks believes, is likely to be in the design of circuits capable of self-repair or the wiring of any specified type of new circuit. No one can say how significant the theory may eventually become for the physical and biological sciences. It may have important bearings on cell growth in embryos, the replication of DNA molecules, the operation of nerve nets, genetic changes in evolving populations and so on. Analogies with life processes are impossible to resist. If a primordial broth of amino acids is large enough, and there is sufficient time, self-replicating, moving automata may result from complex transition rules built into the structure of matter and the laws of nature. There is even the possibility that space-time itself is granular, composed of discrete units, and that the universe, as Fredkin and others have suggested, is a cellular automaton run by an enormous computer. If so, what we call motion may be only simulated motion. A moving spaceship, on the ultimate microlevel, may be essentially the same as one of Conway's spaceships, appearing to move on the macrolevel whereas actually there is only an alteration of states of basic space-time cells in obedience to transition rules that have not yet been discovered.

To the corrections given last month for the original discussion of Conway's game the following should be added. An orthogonal row of eight adjacent counters ends as four blocks and four beehives, and the 5–5–5 row ends as four blocks and two blinkers. In November's illustration of a large spaceship escorted by two smaller ones, the top ship should have been separated by three cells from the middle one. Readers too numerous to mention have confirmed all three corrections.

The number puzzles posed last month by Dr. Matrix have the following answers. In addition to 9,240, the only other number smaller than 10,000 that has 63 proper divisors (including 1 but not the number itself) is 7,560. No other number of four or fewer digits has this many proper divisors. Here is the only way to arrange two three-digit square numbers in a two-by-three matrix so that each column, read from the top down, is a two-digit square:

841
196 .

A newsletter, *The Soma Addict,* is now available at no charge on request from Parker Brothers, P.O. Box 900, Salem, Mass. 01970.



How 13 gliders (color) crash to form a glider gun (generation 75) that oscillates with a period of 30, firing a glider in each cycle

1026556691

LIFE AND DEATH ON A COMPUTER SCREEN

Using a game-like mathematical concept known as cellular automata, some brash young scientists are trying to find the rules that govern one of nature's most profound mysteries

BY CARLA REITER

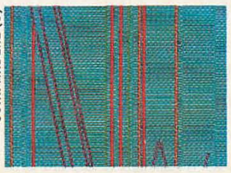
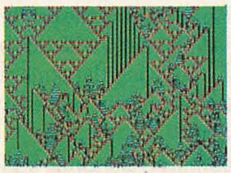
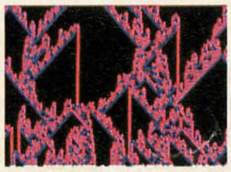
It looks like a video game gone slightly berserk. One line of squares—some green, some blue—forms across the computer screen, then quickly gives birth to another directly beneath it, with the colored squares in different order. This in turn yields still another line. As the new lines appear, one beneath the other, like slats of a slowly descending venetian blind, the random pattern of squares becomes more complex. Some squares turn into a field of triangles. Others form meandering, ivy-like, violet tendrils. In one corner, red and green squares intertwine to become a braid.

"Looks like a class four, rather a pretty one," says the rumpled young man at the computer terminal. Stephen Wolfram is not a video game addict but a scientist in the midst of an ambitious quest. The emerging pattern on the screen is a simple yet powerful mathematical tool called a cellular automaton. That is the forbidding name for arrays of squares—or "cells"—that operate under certain simple rules, generation after generation, interacting with neighboring cells to die or to live and reproduce.

To mathematicians, cellular automata are fascinating abstractions. But to Wolfram, 25, a physicist at the Institute for Advanced Study in Princeton, New Jersey, they are much more. They are helping him look for fundamental laws governing growth and evolution in nature. He has already found hints of such an ordering principle with the discovery that the development of the myriad patterns seems to occur in certain broad categories, or classes. Says Wolfram of his work, "It's kind of a grandiose thing to do."

Grandiose indeed. From filigreed snowflakes that grow from random bits of ice to spiraling mollusc shells that evolve from a few organic molecules, the natural world abounds with complex structures that grow from simple parts. There is probably no better example than life itself, which evolved from a handful of primitive cells into the rich diversity of species on the earth today. But under what mysterious direction did these cells organize themselves into forms of such astonishing complexity?

LAWRENCE BARNS




JOHN MADERE (3)

Physicist Stephen Wolfram ponders his electronic universes in his Princeton office. The screens below show examples of various classes of automata.

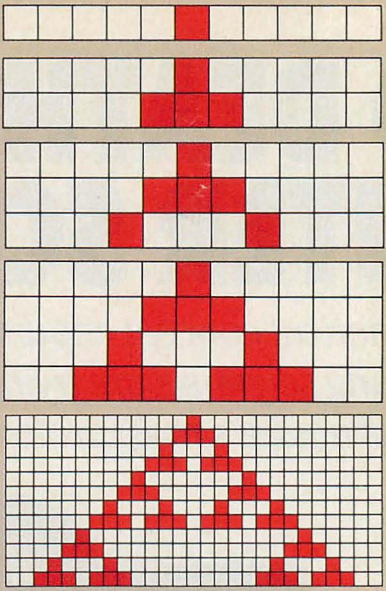
Scientists still do not have a satisfactory answer. All self-organizing systems, as scientists call them, even a snowflake or an amoeba, are so intricate that the general laws governing their growth are buried under layers of obscuring detail. Says Norman Packard, 30, Wolfram's colleague, "You can't even write down the equations for biological evolution because there is just too much going on." And even when the equations for growth are known, as in the case of nonbiological systems like crystallizing snowflakes, they provide no insight into how the overall form develops. Yet scientists are convinced that some universal governing principle underlies the evolution of the multiplicity of forms in nature.

In search of this principle, Wolfram and Packard turned to cellular automata—something they could not have done without powerful computers to per-

DO-IT-YOURSELF AUTOMATON

The diagrams at the right show a simple cellular automaton created by the following rule: as each new row is added to the growing pattern, a square is colored if one, and only one, of the three squares immediately above it in the previous row (either the one directly overhead or one of the two adjacent to it) is also colored. Otherwise, the new square is left blank. As a starting condition, the automaton has only a single row with one colored square. The next generation, or line below, then has three colored squares, the third row two colored squares, and so forth.

You can generate this simple pattern on your own computer with the following Basic language program. The program was written for the IBM PC but should run, with slight modification, on almost any other personal machine.



```
 10 DEFINT A-C,R,X
 20 DIM ROW(80)
 30 ROW(40)=1
 40 SQ$(0)=" ":SQ$(1)=CHR$(178)
 50 B=ROW(0):C=ROW(1)
 60 FOR X=1 TO 79
 70 PRINT SQ$(ROW(X));
 80 A=B:B=C:C=ROW(X+1)
 90 IF (A+B+C)=1 THEN
    ROW(X)=1 ELSE ROW(X)=0
100 NEXT X
110 PRINT
120 GOTO 50
```

To make the automaton more complex, try changing the program by varying the starting condition, altering the rules, or printing squares in different colors.

form the prodigious number of calculations required to investigate the growth of automata over millions and millions of generations. Using these machines, they create what Packard calls "little toy universes," in which they explore the way complicated, ordered structures can grow out of initial conditions of disorder. Says Packard, "We're looking at really simple possibilities—idealized examples—that display this self-organizing behavior. Then we can look at real systems and pull out the features that are doing the organizing."

Thomaso Toffoli, a computer scientist at the Massachusetts Institute of Technology, believes cellular automata are so mathematically powerful that they could supplant the unwieldy differential equations that have been used to solve the problems of physics since Isaac Newton's day, and has designed a special-purpose computer to investigate them. Unlike the Princeton machines, which generate patterns at the pace of a slow-motion movie, Toffoli's computer, a veritable Maserati of its kind, creates generation after generation faster than the blink of an eye.

The invention of cellular automata is generally credited to John von Neumann, the Hungarian-born mathematical genius and pioneering computer theorist who also worked at Princeton. Like Wolfram and Packard after him, he wanted to find a mathematical basis for biology. At the suggestion of Stanislaw Ulam, another émigré mathematician (from Poland), von Neumann devised what was in effect a mathematical analogue, or model, of life. All complicating chemistry was stripped away and only the most essential features—the ability to grow and reproduce—were preserved.

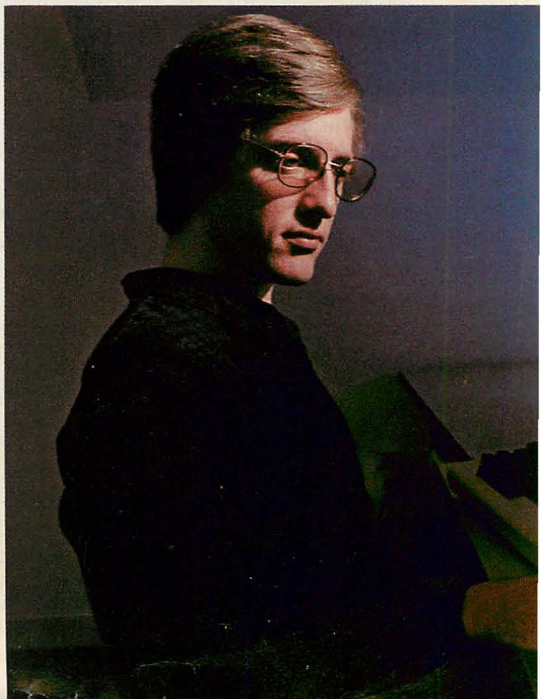
To understand a cellular automaton, imagine it as a game in which people are the cells. At the start, the players arrange themselves in a line. Some sit down, some stand, in no particular order. "There are only two possible moves in this game," say the rules. "You can sit down, or you can stand up. You will decide which you do by looking at the two people on either side of you. If both are sitting or standing, you sit. If one is standing, you stand. Everybody decides what to do and, when you hear the signal, you make your moves at once." The players check their neighbors' status and, at the whistle, assume the appropriate position, changing the configuration of the line. At the next whistle they repeat the action, following the same rules. Each whistle signals a new generation and a new pattern.

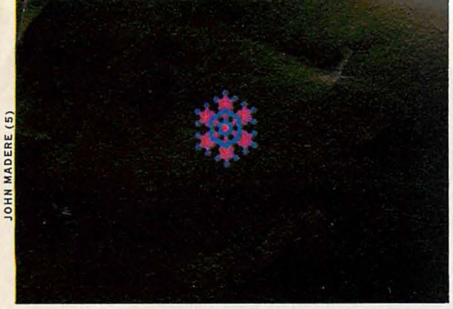
On a computer, the game is played hundreds or thousands of times, each round starting from the arrangement that resulted from the previous turn. Cells—squares in a horizontal line on the computer screen—represent the players. After each round of play, the new configuration of cells is added below those formed in previous rounds, thus preserving the automaton's life history. When the screen is filled, it will scroll upward to make more room.

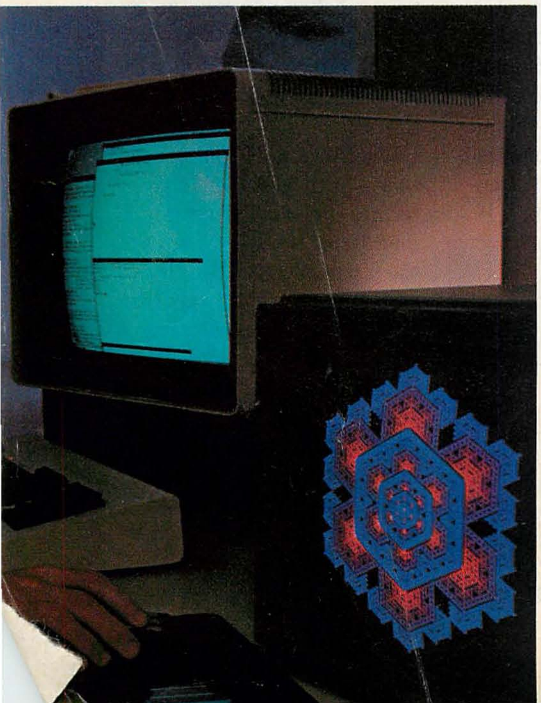
Von Neumann managed to outline some pathways for future research with cellular automata, but his promising work was cut short by his death in 1957 at the age of 54 from cancer—itself a kind of self-reproducing automaton. Still, even in von Neumann's day, it was clear that any experiments with cellular automata would require computers with enormous memories and high speed. Though the course of the generations can be charted with home computers (see box), or even on graph paper, only such machines are able to churn out a sufficient number of generations for any meaningful statistical analysis.

Even so, cellular automata can be a test

Norman Packard observes the growth of his snowflake automata, which are shown, at right, in various stages of development.







for today's most powerful computers. A single rule can generate a bewildering variety of life histories if the starting lineup of cells is changed. For some elaborate automata, the number of possible rules for determining the life or death of cells is greater than the number of particles in the universe (10^{80}). Yet in all the diversity that he has seen played out on his computer screens, Wolfram has spotted signs of some general principles at work.

"You begin to see there are really only four different kinds of things that happen," he says, "and those can be characterized visually in an extremely simple way." He has used these visual clues to postulate four different classes of rules governing cellular automata. One class produces patterns that die out after a few rounds of play. Another builds structures that swing endlessly back and forth between two patterns. A third grows into baroque, apparently chaotic, tangles, and a fourth class—the most intriguing—produces a potpourri of structures that includes elements of the other three classes.

But before either Wolfram or Packard can discern any principles that may underlie the generation of these patterns, they must deal with an obstacle that has stymied every investigator of self-organizing behavior so far. That is the nature of complexity itself. While it is intuitively clear that a seashell is more complicated than, say, the period at the end of this sentence, no one—not even von Neumann—has been able to provide a definition of complexity that is mathematically rigorous and broad enough to be universally applicable. As Packard explains, "If you want to understand how to grow something that is more and more complicated, you have to know what 'more and more complicated' means."

Wolfram defines complexity in terms of idealized computers called, in mathematical jargon, finite state machines. By this reasoning, a simple class of patterns is one produced by a simpler computer than that required to produce a more complicated class. That may seem like a dodge to get around an insurmountable hurdle, but in fact it provides him with a rigorous and flexible definition of complexity. By establishing the size of the smallest computer capable of generating a particular class of patterns, Wolfram is able to get a measure of the pattern's complexity; this gives him a base from which to study the evolution of the complex forms on his screen.

Packard studies snowflakes. These are six-sided figures that grow from a simple "seed," or icy nucleus, into a seemingly infinite variety of forms. Like von Neumann's models of life, Packard's computerized snowflakes preserve only the key features of growth and evolution. His creations begin life as a small cluster of hexagonal cells in the center of the tube, which gradually spread outward in all directions until the screen fills. As a computerized snowflake grows, only the cells on its periphery influence the next move. In essence, the edge of the snowflake is a one-dimensional automaton, like Wolfram's descending strings of cells.

There is obviously a world of difference between nature and cellular automata. The rules that control the growth of Packard's "snowflakes," for example, do not include any details of chemistry, or the thermodynamic laws that govern the transfer of heat and energy in snowflakes created by nature. But the work with cellular automata is only in its infancy. Wolfram, for one, is sure that it will lead to greater understanding of what governs the evolution of complex structures. "The way to find those general laws is not to wallow around in generalities," he says, "but to look at a specific system and work out what it does in detail. Then you can try to figure out what the general laws are likely to look like."

MIT's Tommaso Toffoli stands watch over his specially designed automata machine.

Finding those laws will not be easy. In its infinity of shapes and colors, the world of cellular automata obviously offers an almost unlimited number of systems to compare and study. What remains to be seen is whether the glowing phosphor universes on Wolfram and Packard's computer screens reveal some ultimate truths about the real one. □

Cellular automata: towards a paradigm for complexity

Stephen Wolfram*
The Institute for Advanced Study, Princeton NJ 08540.

January 1984

Cellular automata are discussed as examples of systems in which many simple components act together to produce complex behaviour. They are analysed both as discrete dynamical systems, and as information processing systems. Several universal features are identified, and some general principles are suggested.

It is common in nature to find systems whose overall behaviour is extremely complex, yet whose fundamental component parts are each very simple. The complexity is generated by the cooperative effect of many simple identical components. Much has been discovered about the nature of the components in physical and biological systems; little is known about the mechanisms by which these components act together to give the overall complexity observed. What is now needed is a general mathematical theory to describe the nature and generation of complexity.

Cellular automata are examples of mathematical systems constructed from many identical components, each simple, but together capable of complex behaviour. From their analysis, one may on the one hand develop specific models for particular systems, and on the other hand hope to abstract general principles applicable to a wide variety of complex systems. This article gives an outline of some recent results on cellular automata; more extensive accounts and references may be found in refs. [1-4].

A one-dimensional cellular automaton consists of a line of sites, with each site carrying a value 0 or 1 (or in general $0, \cdots, k-1$). The value a_i of a site at position i is updated in discrete time steps according to an identical deterministic rule depending on a neighbourhood of sites around it:

$$a_i^{(t+1)} = \phi[a_{i-r}^{(t)}, a_{i-r+1}^{(t)}, \cdots, a_{i+r}^{(t)}] \quad . \tag{1}$$

Even with $k=2$ and $r=1$ or 2, the overall behaviour of cellular automata constructed in this simple way can be extremely complex.

Consider first the patterns generated by cellular automata evolving from simple "seeds"

* Work supported in part by the U.S. Office of Naval Research under contract number N00014-80-C-0657.

consisting of a few nonzero sites. Some local rules ϕ give rise to simple behaviour; others produce complicated patterns. An extensive empirical study suggests that the patterns take on four qualitative forms, illustrated in figure 1:

1. Disappears with time.
2. Evolves to a fixed finite size.
3. Grows indefinitely at a fixed speed.
4. Grows and contracts irregularly.

Patterns of the third class are often found to be self-similar or scale invariant. Parts of such patterns, when magnified, are indistinguishable from the whole. The patterns are characterized by a fractal dimension [5]; the value $\log_2 3 \simeq 1.59$ is most commonly found. Many of the self-similar patterns seen in natural systems may in fact be generated by cellular automaton evolution.

Figure 2 shows the evolution of cellular automata from initial states where each site is assigned each of its k possible values with an independent equal probability. Self-organization is seen: ordered structure is generated from these disordered initial states, and in some cases considerable complexity is evident.

Different initial states with a particular cellular automaton rule yield patterns that differ in detail, but are similar in form and statistical properties. Different cellular automaton rules yield very different patterns. An empirical study nevertheless suggests that four qualitative classes may be identified, yielding four characteristic limiting forms:

1. Spatially homogeneous state.
2. Sequence of simple stable or periodic structures.
3. Chaotic aperiodic behaviour.
4. Complicated localized structures, some propagating.

All cellular automata within each class, regardless of the details of their construction and evolution rules, exhibit qualitatively similar behaviour. Such universality should make general results on these classes applicable to a wide variety of systems modelled by cellular automata.

Current mathematical models of natural systems are usually based on differential equations which describe the smooth variation of one parameter as a function of a few others. Cellular automata provide alternative and in some respects complementary models, describing the discrete evolution of a large number of (identical) components. Models based on cellular automata are typically most appropriate in highly non-linear regimes of physical systems, and in chemical and biological systems where discrete thresholds occur. Cellular automata are particularly suitable as models when growth inhibition effects are important.

As one example, cellular automata provide global models for the growth of dendritic crystals (such as snowflakes) [6]. Starting from a simple seed, sites with values representing the solid phase are aggregrated according to a two-dimensional rule that accounts for the inhibition of growth near newly-aggregrated sites, resulting in a fractal pattern of growth. Non-linear chemical reaction-diffusion systems give another example [7]: a simple cellular automaton rule with growth inhibition captures the essential features of the usual partial differential equations, and reproduces the spatial patterns seen. Turbulent fluids may also potentially be modelled as cellular automata with local interactions between discrete vortices on lattice sites.

If probabilistic noise is added to the time evolution rule (1), then cellular automata may be identified as generalized Ising models [8,9]. Phase transitions may occur if ϕ retains some deterministic components, or in more than one dimension.

Cellular automata may serve as suitable models for a wide variety of biological systems. In particular, they may suggest mechanisms for biological pattern formation. For example, the patterns of pigmentation found on many mollusc shells bear a striking resemblance to patterns generated by class 2 and 3 cellular automata (cf. [10]), and cellular automaton models for the growth of some pigmentation patterns have been constructed [11].

Rather than describing specific applications of cellular automata, this article concentrates on general mathematical features of their behaviour. Two complementary approaches provide characterizations of the four classes of behaviour seen in figure 2.

In the first approach, cellular automata are viewed as discrete dynamical systems (e.g. [12]), or discrete idealizations of partial differential equations. The set of possible (infinite) configurations of a cellular automaton forms a Cantor set; cellular automaton evolution may then be viewed as a continuous mapping on this Cantor set. Quantities such as entropies, dimensions and Lyapunov exponents may then be considered for cellular automata.

In the second approach, cellular automata are instead considered as information-processing systems (e.g. [13]), or parallel-processing computers of simple construction. Information represented by the initial configuration is processed by the evolution of the cellular automaton. The results of this information processing may then be characterized in terms of the types of formal languages generated. (Notice that the mechanisms for information processing in natural system appear to be much closer to those in cellular automata than in conventional serial-processing computers: cellular automata may therefore provide efficient media for practical simulations of many natural systems.)

Most cellular automaton rules have the important feature of irreversibility: several different configurations may evolve to a single configuration, and with time a contracting subset of all possible configurations appears. Starting from all possible initial configurations, the cellular automaton evolution may generate only special "organized" configurations, and "self-organization" may occur.

For class 1 cellular automata, essentially all initial configurations evolve to a single final configuration, analogous to a limit point in a continuous dynamical system. Class 2 cellular automata evolve to limit sets containing essentially only periodic configurations, analogous to limit cycles. Class 3 cellular automata yield chaotic aperiodic limit sets, containing analogues of chaotic or "strange" attractors.

Entropies and dimensions give a generalized measure of the density of the configurations generated by cellular automaton evolution. The (set) dimension or limiting (topological) entropy for a set of cellular automaton configurations is defined as (e.g. [12])

$$d^{(x)} = \lim_{X\to\infty} \frac{1}{X}\log_k N(X)\,, \tag{2}$$

where $N(X)$ gives the number of distinct sequences of X site values that appear. For the set of possible initial configurations, $d^{(x)}=1$. For a limit set containing only a finite total number of configurations, $d^{(x)}=0$. For most class 3 cellular automata, $d^{(x)}$ decreases with time, giving, $0<d^{(x)}<1$, and suggesting that a fractal subset of all possible configurations occurs.

A dimension or limiting entropy $d^{(t)}$ corresponding to the time series of values of a single site may be defined in analogy with (2). (The analogue of (2) for a sufficiently wide patch of sites yields a topologically-invariant entropy for the cellular automaton mapping.) $d^{(t)}=0$ for periodic sets of configurations.

$d^{(x)}$ and $d^{(t)}$ may be modified to account for the probabilities of configurations by defining

$$d_\mu^{(x)} = -\lim_{X\to\infty} \frac{1}{X}\sum_{j=1}^{k^X} p_j \log_k p_j\ , \tag{3}$$

and its analogue, where p_j are probabilities for possible length X sequences. These measure dimensions may be used to delineate the large time behaviour of the different classes of cellular automata:

1. $d_\mu^{(x)}=d_\mu^{(t)}=0$.
2. $d_\mu^{(x)}>0$, $d_\mu^{(t)}=0$.

3. $d_\mu^{(s)}>0$, $d_\mu^{(t)}>0$.

As discussed below, dimensions are usually undefined for class 4 cellular automata.

Cellular automata may also be characterized by the stability or predictability of their behaviour under small perturbations in initial configurations. Figure 3 shows differences in patterns generated by cellular automata resulting from a change in a single initial site value. Such perturbations have characteristic effects on the four classes of cellular automata:

1. No change in final state.
2. Changes only in a finite region.
3. Changes over an ever-increasing region.
4. Irregular changes.

In class 1 and 2 cellular automata, "information" associated with site values in the initial state propagates only a finite distance; in class 3 cellular automata, it propagates an infinite distance at a fixed speed, while in class 4 cellular automata, it propagates irregularly, but over an infinite range. The speed of information propagation is related to the Lyapunov exponent for the cellular automaton evolution, and measures the degree of sensitivity to initial conditions (cf. [14]). It leads to different degrees of predictability for the outcome of cellular automaton evolution:

1. Entirely predictable, independent of initial state.
2. Local behaviour predictable from local initial state.
3. Behaviour depends on an ever-increasing initial region.
4. Behaviour effectively unpredictable.

Information propagation is particularly simple for the special class of additive cellular automata (whose local rule function ϕ is linear modulo k), in which patterns generated from arbitrary initial states may be obtained by superposition of patterns generated by evolution of simple initial states containing a single non-zero site. A rather complete algebraic analysis of such cellular automata may be given [15]. Most cellular automata are not additive; however, with special initial configurations it is often possible for them to behave just like additive rules. Thus for example the evolution of an initial configuration consisting of a sequence of 00 and 01 digrams under one rule may be identical to the evolution of the corresponding "blocked" configuration consisting of 0 and 1 under another rule. In this way, one rule may simulate another under a blocking transformation (analogous to a renormalization group transformation). Evolution from an arbitrary initial state may be attracted to (or repelled from) the special set of configurations for which such a simulation occurs. Often several phases exist, corresponding to different blocking transformations: sometimes phase boundaries move at constant speed, and one phase rapidly takes over; in other cases, phase boundaries execute random walks, annihilating in pairs, and leading to a slow increase in the average domain size [16,17], as illustrated in figure 4. Many rules appear to follow attractive simulation paths to additive rules, which correspond to fixed points of blocking transformations. The behaviour of many rules at large times and on large spatial scales is thus determined by the behaviour of additive rules.

Decreases with time in the spatial entropies and dimensions of eqns. (2) and (3) signal irreversibility in cellular automaton evolution. Some cellular automaton rules are however reversible, so that each every configuration has a unique predecessor in the evolution, and the spatial entropy and dimension of eqns. (2) and (3) remain constant with time. Figure 5 shows some examples of the evolution of such rules, constructed by adding a term $-a_i^{(t-1)}$ to eqn. (1) [18]. Once again, there are analogues of the four classes of behaviour seen in figure 2, distinguished by the range and speed of information propagation.

Conventional thermodynamics gives a general description of systems whose microscopic evolution is reversible; it may therefore be applied to reversible cellular automata such as those of figure 4. As usual, the "fine-grained" entropy for sets (ensembles) of configurations, computed as in eqn. (3) with perfect knowledge of each site value, remains constant in time. The "coarse-grained" entropy for configurations is nevertheless almost always non-decreasing with time, as required by the second law of thermodynamics. Coarse-graining emulates the imprecision of practical measurements, and may be implemented by applying almost any contractive mapping to the configurations (a few iterations of an irreversible cellular automaton rule suffice). For example, coarse-grained entropy might be computed by applying eqn. (3) to every fifth site value. In an ensemble with low coarse-grained entropy, the values of every fifth site would be highly constrained, but arbitrary values for the intervening sites would be allowed. Then in the evolution of a class 3 or 4 cellular automaton the disorder of the intervening site values would "mix" with the fifth site values, and the coarse-grained entropy would tend towards its maximum value. Signs of self-organization in such systems must be sought in temporal correlations, often manifest in "fluctuations" or metastable "pockets" of order.

While all fundamental physical laws appear to be reversible, macroscopic systems often behave irreversibly, and are appropriately described by irreversible laws. Thus, for example, although the microscopic molecular dynamics of fluids is reversible, the relevant macroscopic velocity field obeys the irreversible Navier-Stokes equations. Conventional thermodynamics does not apply to such intrinsically irreversible systems: new general principles must be found. Thus for cellular automata with irreversible evolution rules, coarse-grained entropy typically increases for a short time, but then decreases to follow the fine-grained entropy. Measures of the structure generated by self-organization in the large time limit are usually affected very little by coarse-graining.

Quantities such as entropy and dimension, suggested by information theory, give only rough characterizations of cellular automaton behaviour. Computation theory suggests more complete descriptions of self-organization in cellular automata (and other systems). Sets of cellular automaton configurations may be viewed as formal languages, consisting of sequences of symbols (site values) forming words according to definite grammatical rules (e.g. [19]). The set of all possible initial configurations corresponds to a trivial formal language. The set of configurations obtained after any finite number of time steps are found to form a regular language. The words in a regular language correspond to the possible paths through a finite graph representing a finite state machine. It can be shown that a unique smallest finite graph reproduces any given regular language. Examples of such graphs are shown in figure 6. These graphs give complete specifications for sets of cellular automaton configurations (ignoring probabilities). The number of nodes Ξ in the smallest graph corresponding to a particular set of configurations may be defined as the "regular language complexity" of the set. It specifies the size of the minimal description of the set in terms of regular languages. Larger Ξ correspond to more complicated sets. (Notice that the topological entropy of a set is given by the logarithm of the algebraic integer obtained as the largest root of the characteristic polynomial for the incidence matrix of the corresponding graph. The characteristic polynomials for the graphs in fig. 5 are $2-\lambda$ $(\lambda_{\max}=2)$, $1-\lambda+2\lambda^2-\lambda^3$ $(\lambda_{\max}\simeq 1.755)$ and $-1+\lambda-\lambda^2+2\lambda^3-4\lambda^4+\lambda^5+3\lambda^6-5\lambda^7+3\lambda^8-3\lambda^9+5\lambda^{10}-6\lambda^{11}+4\lambda^{12}-\lambda^{13}$ $(\lambda_{\max}\simeq 1.732)$, respectively.)

It appears that the regular language complexity Ξ for sets generated by cellular automaton evolution is almost always non-decreasing with time. Increasing Ξ signals increasing self-organization. Ξ may thus represent a fundamental property of self-organizing systems, complementary to entropy. It may in principle be extracted from experimental data.

Cellular automata that exhibit only class 1 and 2 behaviour always appear to yields sets that correspond to regular languages in the large time limit. Class 3 and 4 behaviour typically gives rise, however, to a rapid increase of Ξ with time, presumably leading to limiting sets not described by regular languages.

Formal languages are recognized or generated by idealized computers with a "central processing unit" containing a fixed finite number of internal states, together with a "memory". Four types of formal languages are conventionally identified, corresponding to four types of computer:

Regular languages: no memory required.

Context-free languages: memory arranged as a last-in first-out stack.

Context-sensitive languages: memory as large as input word required.

Unrestricted languages: arbitrarily large memory required (general Turing machine).

Examples are known of cellular automata whose limiting sets correspond to context-free languages [19]. Arguments can be given that the limit sets for class 3 cellular automata typically form context-sensitive languages, while those for class 4 cellular automata correspond to unrestricted languages. (Notice that while a minimal specification for any regular language may always be found, there is no finite procedure to obtain a minimal form for more complicated formal languages: no generalization of the regular language complexity Ξ may thus be given.)

While dynamical systems theory concepts suffice to define class 1, 2 and 3 cellular automata, computation theory is apparently required for class 4 cellular automata. Examples of the evolution of a typical class 4 cellular automaton are shown in figure 7. Varied and complicated behaviour, involving many different time scales, is evident. Persistent structures are often generated; the smallest few are illustrated in figure 8, and are seen to allow both storage and transmission of information. It seems likely that the structures supported by this and other class 4 cellular automata rule may be combined to implement arbitrary information processing operations. Class 4 cellular automata would then be capable of universal computation: with particular initial states, their evolution could implement any finite algorithm. (Universal computation has been proved for a $k=18$, $r=1$ rule [20], and for two-dimensional cellular automata such as the "Game of Life" [21].) A few percent of cellular automaton rules with $k>2$ or $r>1$ are found to exhibit class 4 behaviour: all these would then in fact be capable of arbitrarily complicated behaviour. This capability precludes a smooth infinite size limit for entropy or other quantities: as the size of cellular automaton considered increases, more and more complicated phenomena may appear.

Cellular automaton evolution may be viewed as a computation. Effective prediction of the outcome of cellular automaton evolution requires a short-cut that allows a computation more efficient than the evolution itself. For class 1 and 2 cellular automata, such short-cuts are clearly possible: simple computations suffice to predict their complete future. The computational capabilities of class 3 and 4 cellular automata may however be sufficiently great that in general they allow no short-cuts. The only effective way to determine their evolution from a given initial state would then be by explicit observation or simulation: no finite formulae for their general behaviour could be given. (If class 4 cellular automata are indeed capable of universal computation, then the variety of their possible behaviour would preclude general prediction, and make explicit observation or simulation necessary.) Their infinite time limiting behaviour could then not in general be determined by any finite computational process, and many of their limiting properties would be formally undecidable. Thus, for example, the "halting problem" of determining whether a class 4 cellular automaton with a given finite initial configuration ever evolves to the null configuration would be undecidable. An explicit simulation could determine only whether halting occurred before some fixed time, and no whether it occurred after an arbitrarily long time.

It seems likely that for class 4 cellular automata, the outcome of evolution from almost all initial configurations can be determined only by explicit simulation, while for class 3 cellular automata this is the case for only a small fraction of initial states. Nevertheless, this possibility suggests that the occurrence of particular site value sequences in the infinite time limit is in general undecidable. The large time limit of the entropy for class 3 and 4 cellular automata would then in general be non-computable: bounds on it could be given, but there could be no finite procedure to compute it to arbitrary precision. (This would be the case if the limit sets for class 3 and 4 cellular automata formed at least context-sensitive languages.)

While the occurrence of a particular length n site value sequence in the infinite time limit may be undecidable, its occurrence after any finite time t can in principle be determined by considering all length $n_0=n+2rt$ initial sequences that could evolve to it. For increasing n or t this procedure would nevertheless involve exponentially-growing computational resources, so that it would rapidly become computationally intractable. It seems likely that the identification of possible sequences generated by class 3 and 4 cellular automata is in general an NP-complete problem (e.g. [13]). It can therefore presumably not be solved in any time polynomial in n or t, and essentially requires explicit simulation of all possibilities.

Undecidability and intractability are common in problems of mathematics and computation. They may well afflict all but the simplest cellular automata. One may speculate that they are widespread in natural systems, perhaps occurring almost whenever nonlinearity is present. No simple formulae for the behaviour of many natural systems could then be given; the consequences of their evolution could be found effectively only by direct simulation or observation.

I am grateful to O. Martin, J. Milnor, N. Packard and many others for discussions. The computer mathematics system SMP [22] was used in the course of this work.

References

1. S. Wolfram, "Statistical mechanics of cellular automata", Rev. Mod. Phys. 55 (1983) 601.
2. S. Wolfram, "Universality and complexity in cellular automata", Physica D, to be published.
3. S. Wolfram, "Computation theory of cellular automata", Institute for Advanced Study preprint (January 1984).
4. S. Wolfram, "Cellular automata", Los Alamos Science, Fall 1983 issue.
5. B. Mandelbrot, "The fractal geometry of nature", Freeman (1982).
6. N. Packard, "Cellular automaton models for dendritic growth", Institute for Advanced Study preprint, in preparation.
7. B. Madore and W. Freedman, "Computer simulations of the Belousov-Zhabotinsky reaction", Science 222 (1983) 615.
8. G. Vichniac, "Simulating physics with cellular automata", Physica D, to be published.
9. E. Domany and W. Kinzel, "Equivalence of cellular automata to Ising models and directed percolation", Stanford University preprint (November 1983).
10. C. H. Waddington and R. J. Cowe, "Computer simulation of a molluscan pigmentation pattern", J. Theoret. Biol. 25 (1969) 219; D. T. Lindsay, "Simulating molluscan shell pigment lines and states: implications for pattern diversity", Veliger, 24, 297; J. H. Campbell, private communication.
11. D. A. Young, "A local activator-inhibitor model of vertebrate skin patterns", Lawrence Livermore National Laboratory report (February 1983).
12. J. Guckenheimer and P. Holmes, "Nonlinear oscillations, dynamical systems, and bifurcations of vector fields", Springer (1983).
13. J. E. Hopcroft and J. D. Ullman, "Introduction to automata theory, languages, and computation", Addison-Wesley (1979).
14. N. Packard, "Complexity of growing patterns in cellular automata", Institute for Advanced Study preprint (October 1983).
15. O. Martin, A. Odlyzko and S. Wolfram, "Algebraic properties of cellular automata", Comm. Math. Phys., to be published.
16. P. Grassberger, "A new mechanism for deterministic diffusion", Phys. Rev. A, to be published; "Chaos and diffusion in deterministic cellular automata", Physica D, to be published.
17. D. Lind, "Applications of ergodic theory and sofic systems to cellular automata", Physica D, to be published.

18. E. Fredkin and N. Margolus, private communications; N. Margolus, "Physics-like models of computation", Physica D, to be published.

19. D. Hillis and L. Hurd, private communication; L. Hurd, to be published.

20. A. R. Smith, "Simple computation-universal cellular spaces", J. ACM 18 (1971) 331.

21. E. R. Berlekamp, J. H. Conway and R. K. Guy, "Winning ways for your mathematical plays", Academic Press, vol. 2, chap. 25; M. Gardner, "Wheels, Life and other mathematical amusements", Freeman (1983).

22. S. Wolfram, "SMP reference manual", Inference Corporation (Los Angeles) (1983).

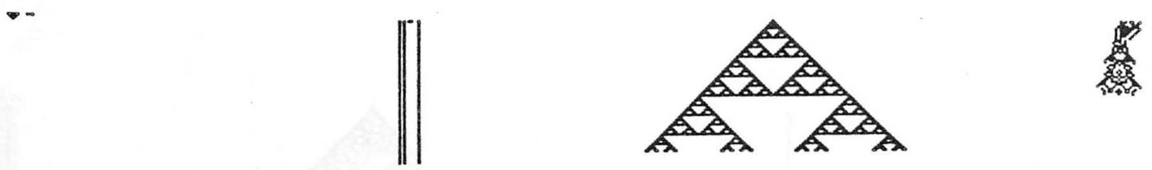


Figure 1: Classes of patterns generated by the evolution of cellular automata from simple "seeds". Successive rows correspond to successive time steps in the cellular automaton evolution. Each site is updated at each time step according to eqn. (1) by cellular automaton rules that depend on the values of a neighbourhood of sites at the previous time step. Sites with values 0 and 1 are represented by white and black squares, respectively. Despite the simplicity of their construction, patterns of some complexity are seen to be generated. The rules shown exemplify the four classes of behaviour found. (The first three are $k=2$, $r=1$ rules with rule numbers [1] 128, 4 and 126, respectively; the fourth is a $k=2$, $r=2$ rule with totalistic code [2] 52.) In the third case, a self-similar pattern is formed.

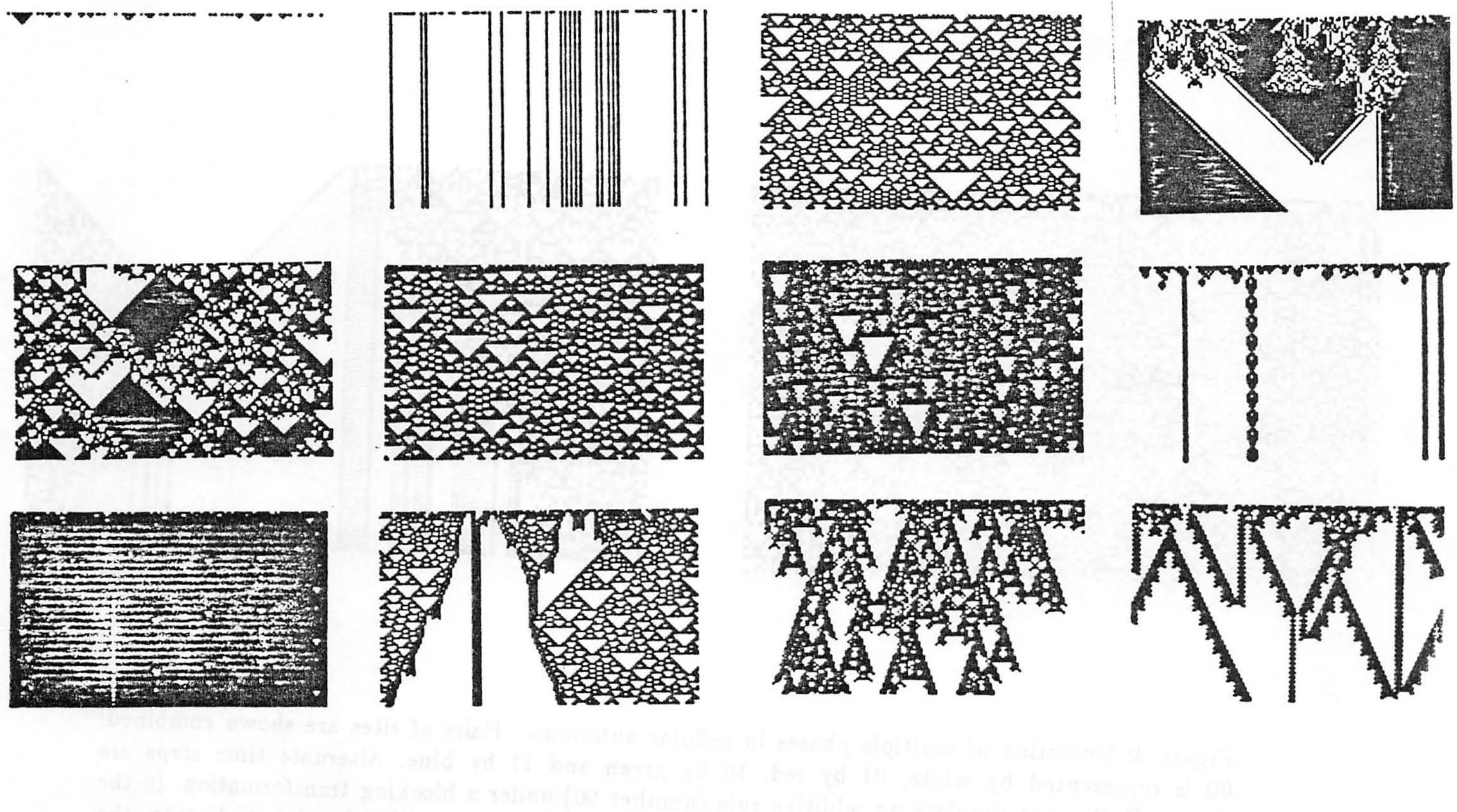


Figure 2: Evolution of various cellular automata from disordered initial states. In many cases, ordered structure is seen to be generated. The black and white pictures show examples of the four qualitative classes of behaviour found. (The rules shown are the same as in figure 1.) The colour pictures show rules with $k=5$ (five possible values for each site) and $r=1$ (nearest neighbour rules). Site values 0 through 4 are represented by white, red, green, blue and yellow squares, respectively. (The rules shown have totalistic codes 10175, 566780, 570090, 580020, 583330, 672900, 5694390, 59123000.)

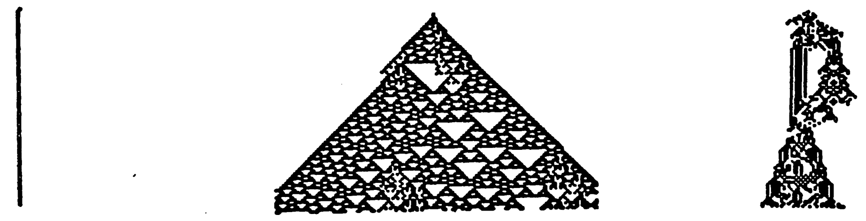


Figure 3: Evolution of small initial perturbations in cellular automata, as shown by the difference (modulo two) between patterns generated from two disordered initial states differing in the value of a single site. The examples shown illustrate the four classes of behaviour found. Information on changes in the initial state almost always propagates only a finite distance in the first two classes, but may propagate an arbitrary distance in the third and fourth classes.

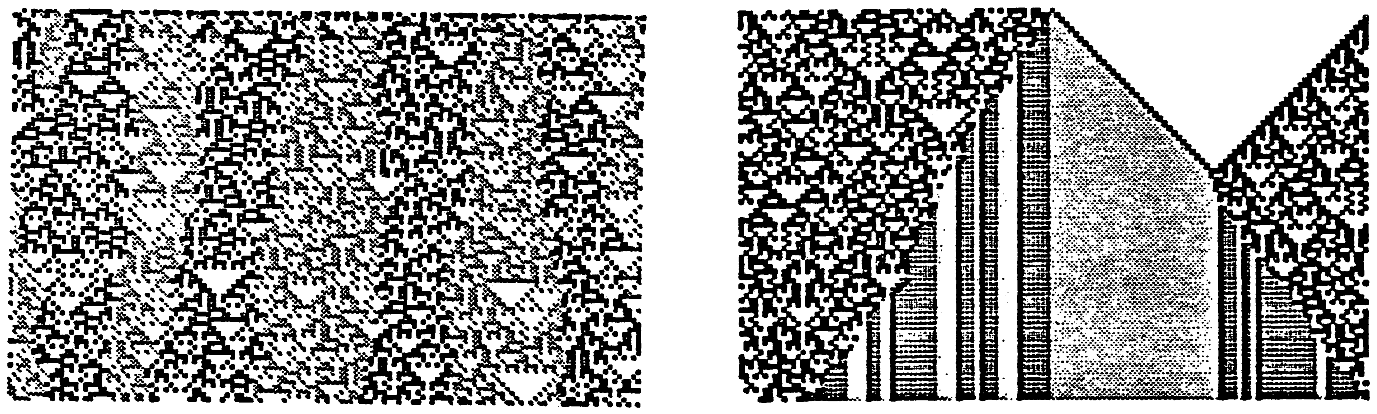


Figure 4: Evolution of multiple phases in cellular automata. Pairs of sites are shown combined: 00 is represented by white, 01 by red, 10 by green and 11 by blue. Alternate time steps are shown. Both rules simulate an additive rule (number 90) under a blocking transformation. In the first rule (number 18), the simulation is attractive: starting from a disordered initial state, the domains grow with time. In the second rule (number 94), the simulation is repulsive: only evolution from a special initial state yields additive rule behaviour; a defect is seen to grow, and attractive simulation of the identity rule takes over.

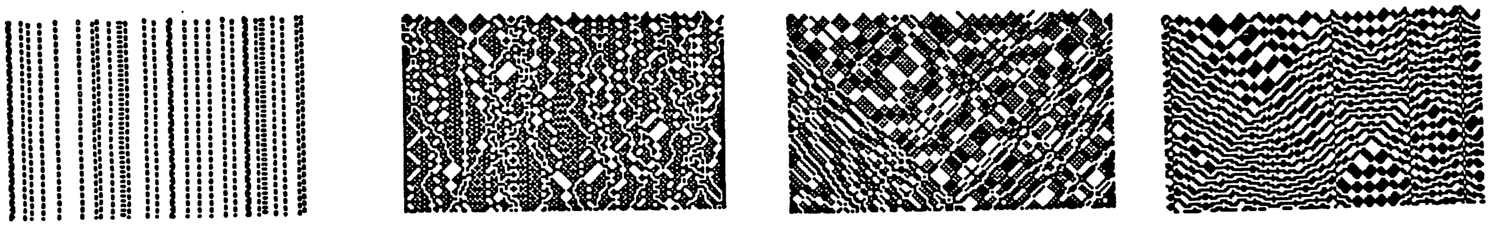


Figure 5: Evolution of some cellular automata with reversible rules. Each configuration is a unique function of the two previous configurations. (Rule numbers 4, 22, 90 and 126 are shown.) As initial conditions, each site in two successive configurations is chosen to have value 1 with probability 0.1.

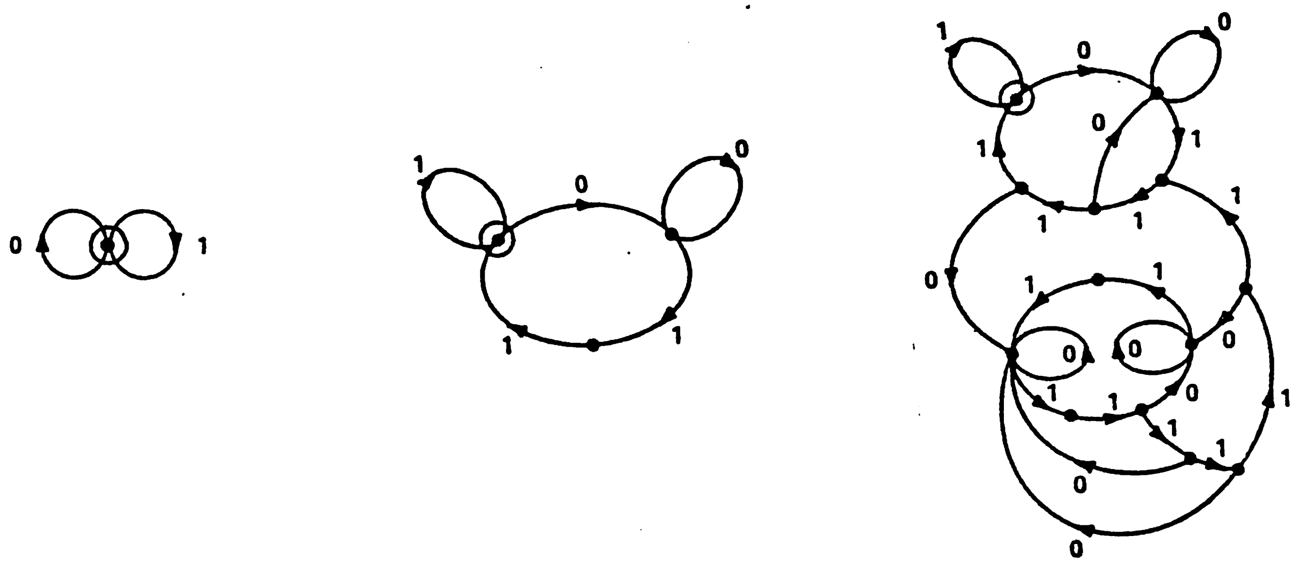


Figure 6: Graphs representing the sets of configurations generated in the first few time steps of evolution according to a typical class 3 cellular automaton rule (k=2, r=1 rule number 126). Possible configurations correspond to possible paths through the graphs, beginning at the encircled node. At t=0, all possible configurations are allowed. With time, a contracting subset of configurations are generated. (After one time step, for example, no configuration containing the sequence of site value 101 can appear.) At each time step, the complete set of possible configurations forms a regular formal language: the graph gives a minimal complete specification of it. The number of nodes in the graph gives a measure of the complexity Ξ of the set, viewed as a regular language. As for other class 3 cellular automata, the complexity of the sets grows rapidly with time; for t=3, Ξ=107, and t=4, $\Xi \gtrsim 4000$.

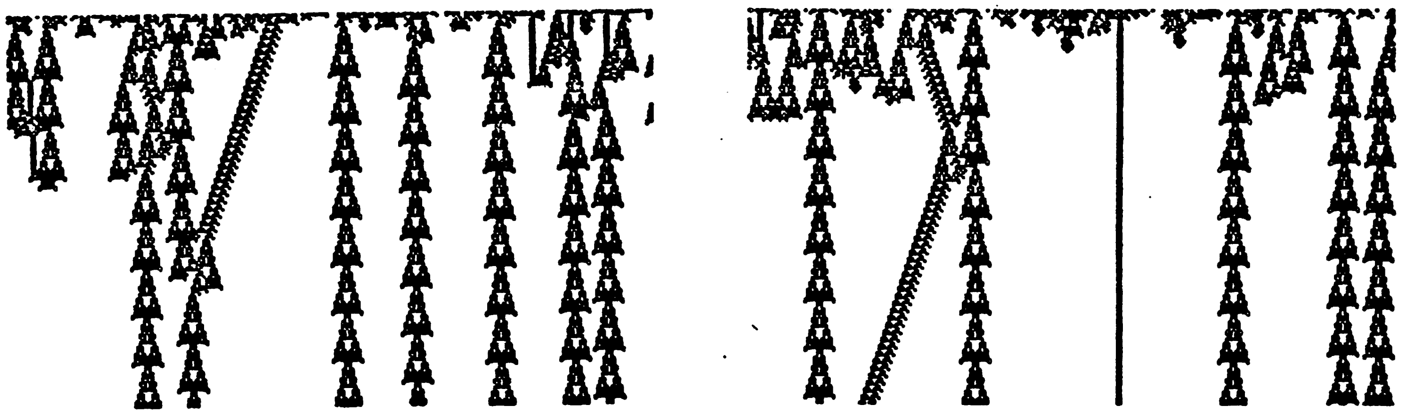


Figure 7: Examples of the evolution of a typical class 4 cellular automaton from disordered initial states. This and other class 4 cellular automata are conjectured to be capable of arbitrary information processing, or universal computation. The rule shown has $k=3$, $r=1$, and takes the value of a site to be 1 if the sum of the values of the sites in the three-site neighbourhood is 2 or 6, to be 2 if the sum is 3, and to be zero otherwise (totalistic code 792).

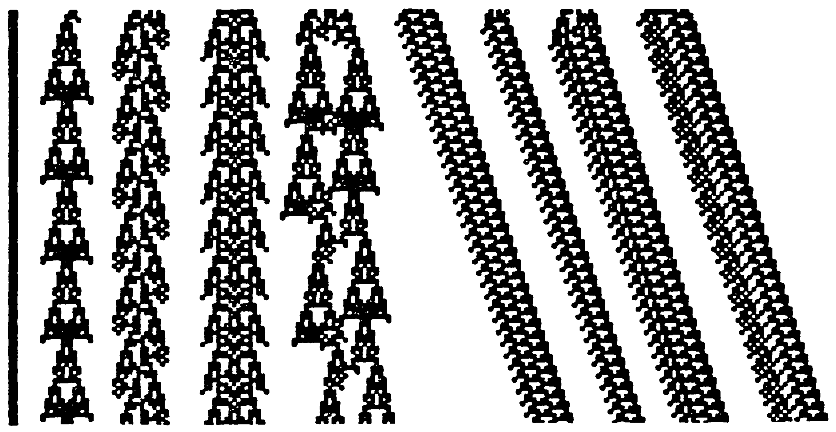


Figure 8: Persistent structures generated in the evolution of the class 4 cellular automaton of figure 7. The first four structures shown have periods 1, 20, 16 and 12 respectively; the last four structures (and their reflections) propagate: the first has period 32, the next three period 3, and the last period 6. These structures are some of the elements required to support universal computation.

1026 55669

Physica 10D (1984) 195–204
North-Holland, Amsterdam

CAM: A HIGH-PERFORMANCE CELLULAR-AUTOMATON MACHINE*

Tommaso TOFFOLI
MIT Laboratory for Computer Science, 545 Technology Sq., Cambridge, MA 02139, USA

CAM is a high-performance machine dedicated to the simulation of cellular automata and other distributed dynamical systems. Its speed is about one-thousand times greater than that of a general-purpose computer programmed to do the same task; in practical terms, this means that CAM can show the evolution of cellular automata on a color monitor with an update rate, dynamic range, and spatial resolution comparable to those of a Super-8 movie, thus permitting intensive interactive experimentation. Machines of this kind can open up novel fields of research, and in this context it is important that results be easy to obtain, reproduce, and transmit. For these reasons, in designing CAM it was important to achieve functional simplicity, high flexibility, and moderate production cost. We expect that many research groups will be able to own their own copy of the machine to do research with.

1. Introduction

A *cellular automation* is a stylized "universe." It consists of a uniform checkerboard, with each square or cell containing a few bits of data; time advances in discrete jumps; and the "laws of the universe" are represented by a single rule – say, a small look-up table – through which at each time-step each cell computes its new state from that of its nearest neighbors. Given a suitable rule, such a simple underlying mechanism is sufficient to support a whole hierarchy of structures, phenomena, and properties. (See enclosed color plates, which are shared by [2], [5] and [6].) Cellular automata provide eminently usable models for many investigations in natural sciences, combinatorial mathematics, and computer science [4, 5].

However, the generality and flexibility of the cellular-automaton model are achieved at a cost. Instead of having relatively few lumped variables that interact in an arbitrarily assigned way, as in many *ad hoc* models of physical phenomena, a cellular automaton uses very many variables (i.e., one per cell) that interact only locally and uniformly. In order to synthesize structures of significant complexity it is necessary to use a large number of cells (typically tens of thousands, or even millions), and in order for these structures to interact with one another and evolve to a significant extent it is necessary to let the automaton run for a large number of time steps (typically, many thousand). Thus, satisfactory experimental runs with cellular automata require performing billions of individual cell-updating operations.

When the simulation of a cellular automaton is carried out on a general-purpose sequential computer, each of these cell-updating operations may require thirty-odd machine instructions, each one involving a few machine cycles (instruction fetch, instruction decode, memory read, etc.), for a total of perhaps one hundred machine cycles, i.e., a time of $\approx 100\,\mu$s. When this is multiplied by a billion, one easily arrives at computer runs lasting $\approx 10^5$ s, or in the order of days!

The most natural and effective way to implement cellular automata is of course as an array of identical VLSI chips each one containing, in turn, an array of thousands of identical cells. In this way, it is possible to arrive at arrays consisting of

* This research was supported in part by the Defense Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contracts Nos. N00014–75–C–0661 and N00014–83–K–0125, and in part by NSF Grant No. 8214312–1ST.

0167-2789/84/$03.00 © Elsevier Science Publishers B.V.
(North-Holland Physics Publishing Division)

millions of cells; since direct communication between cells is strictly local, such arrays could be run at a clock rate of a few nanoseconds. Overall, the computation might proceed 10^6 to 10^9 times faster than on a general-purpose computer. However, no one really knows to what extent this enormous but quite specialized bit-crunching power can be effectively applied to real problems. One would have to experiment with different kinds of rules, different neighborhood structures, etc., as well as develop some theoretical background (cf. Toffoli [5]). And while in the long run the cost per cell of arrays of this kind could be quite low, blindly investing in fabricating one large array of a particular type without knowing whether that is what was actually wanted could be a very expensive proposition. On the other hand, if we tried to study the potential of a particular structure, in advance of fabricating it, by means of general-purpose computer simulation, the simulation would be so many orders of magnitude slower than the target as to fail to reveal the target's capabilities.

The present approach is meant to break the above *impasse*. We still rely on sequential processing (which, as we shall see below, has many advantages), but through special-purpose hardware. A number of ingenious architectural solutions make it possible to gain a speed factor of ≈ 1000 over simulation by general-purpose computer, and this at a cost and circuit complexity comparable to those of inexpensive microcomputer systems. A further factor of ≈ 100 can easily be achieved by applying a bit more of brute force (faster components, interleaving of the transition-table memory, etc.), for an overall factor of $\approx 10^5$. Thus, machines of the kind of CAM, while still much slower than a truly parallel implementation, place a stepping stone across a gap that is perhaps too wide to bridge in one step, and can show us in a time of minutes what a parallel array would do in milliseconds and a more conventional simulation would take weeks to achieve.

2. Overview of CAM

2.1. *Generalities*

To sum up Section 1, it is desirable to have a tool capable of carrying out in an efficient way the immense computational task involved in explicitly tracing a cellular automaton's trajectory. In other words, we would like to be able to choose a particular law, assign an initial state, say "Go!," and see the system evolve under our eyes. At times we may want to slow down the process or single-step through it; examine in detail the current state and possibly save it for further analysis, or as a "seed" from which to restart; introduce perturbations; etc. As the computation proceeds, we may want to perform some statistical analysis on the sequence of states, or set up a trap which will alert us if a particular event has taken place. In other words, we would like to have a fast, flexible, easy-to-use *cellular-automaton simulator*.

CAM is a "black box" containing a high-performance parallel processor optimized toward the simulation of cellular automata and other distributed, discrete systems*. Here we shall summarize the architecture and the functional characteristics of this machine†. Its intented uses are discussed in other papers [5, 2, 6].

Access to the box is via two interface ports. The *user* port is typically connected to a general-purpose computer** through which the user controls the machine; the *monitor* port is connected to a standard RGB color monitor (or a black-and-write CRT monitor), which displays the evolution of the cellular automaton.

Under software control, the user can (a) read or white the memory planes describing the automaton's state, (b) read or write the memory tables

* CAM was conceived and developed by Tommaso Toffoli in 1981, initially as a personal project and on spare time and resources. One version (CAM 1.2) was built in the context of ongoing MIT research, and partially supported by it.

† Special-purpose pseudo-parallel processors based on techniques similar to those used in CAM have been in use for some time [1, 3]. These machines are much more specialized than CAM, and essentially are "one-of-a-kind" research tools.

** A microcomputer with terminal, 32K of RAM, and floppy-disk storage is adequate for many applications.

containing a selection of transition functions and display functions, and (c) request the execution of a parallel step using one of the available transition functions, or (d) read or write miscellaneous registers which specify or reflect the machine's status.

2.2. *Space, time, and state variables*

The "universe" of CAM consists of 65,536 *cells* arranged in a two-dimensional array of size 256 × 256. Each cell may contain up to 8 bits of data – corresponding to 256 distinct cell states. From the user's viewpoint, it will be convenient to visualize the array as a stack of 8 transparencies – or memory *planes* – each contributing a single bit to each cell. The array "wraps around" both vertically and horizontally, i.e., has the structure of a two-dimensional torus; this corresponds to *periodic* boundary conditions*. The whole array is scanned and displayed (and – if a step is requested – updated) 60 times per second, synchronously with the scanning of the CRT monitor.

2.3. *Dynamics*

The dynamics of the system – i.e., the law according to which at each step the whole array is updated by replacing the current value of each cell with a new value – is specified by a *transition function*. This can be hardwired, but is usually encoded in RAM *tables*. Typically, one supplies as arguments to the transition function the current values of the neighbors of the cell to be updated†. This yields a law that is *local* (what happens at any point depends only on what there is in the vicinity of that point) and *uniform* (the law is the same at each point). However, more general dynamical systems may be obtained by introducing other classes of arguments (cf. section 5). In particular, the hardware of CAM supplies a distinct address for each cell, so that one can construct laws that are *space-dependent*; and optionally supplies random values, so that the law can be made *non-deterministic*. Finally, it is possible to use time itself as an argument in order to realize *time-dependent* systems.**

2.4. *Display*

In CAM, one can actually *watch*, in real time, the evolution of the system under study. To enhance observations, instead of using a one-to-one mapping between the state of a cell and the color of the corresponding pixel on the CRT, one can introduce a suitable *display function*. Much as the transition function, the display function may use both local arguments (extracted from the state of the neighborhood) and nonlocal ones. Thus one can "filter out" interesting dynamical patterns, highlight selected areas, and even caption specific events.

2.5. *Functional parallelism vs. sequential implementation*

In a truly *parallel* implementation of cellular automata, all cells are updated at the same time. For this to be possible, each cell must have independent access to the transition-function table, or have its own private copy of it. In practice, the hardware that accesses or embodies the transition function may be enormously larger than that

* Other boundary conditions (e.g., absorbing reflecting, etc.) can be realized by tampering with the transition function in a nonlocal way; cf. below.

† Note that the cell itself is counted among its neighbors. The CAM hardware provides a neighborhood of 3 × 3, i.e., the center cell itself, its 4 nearest neighbors, and its 4 second-nearest neighbors. Larger neighborhoods can be synthesized as explained in section 5.1.

** All of the above extensions can be realized in a variety of ways, and the user will learn to appreciate the convenience of such an arrangement. To give but one example, distinguished places acting as boundaries, sources, or sinks can be defined by decoding the address of the given places and feeding the result as a *nonlocal* argument to the transition function. Alternatively, one can distinguish between places by loading one memory plane with a suitable spatial pattern (which will be kept fixed by the transition function); this pattern will be "sensed" by the other planes in a *local* way – that is, through the neighborhood of each cell – and thus will allow them to change their behavior from place to place.

which realizes a cell's state variables (cf. section 3.1). Thus, for a general-purpose system such as CAM, it is much more convenient to have only one copy of the transition-function hardware, and time-share it between cells. Cell updating is thus sequential. This approach raises a number of problems – the main one being that, as soon as one cell is updated, the next cell will immediately see the new state (rather than the old one it needs in order to compute its own new state). In CAM, this and similar problems are solved by providing cells with short-term memory, achieved through a pipelined architecture. In addition, cells near the edges of the array are also given look-ahead capabilities, in order to realize the correct wrap-around*.

All of the above artifices are totally transparent to the user (and for this reason will not be further discussed here). In conclusion, from a functional viewpoint, CAM behaves in all respects as a *truly parallel system*, and there is an exact isomorphism between the abstract "target" cellular automaton and its electronic realization.

Of course, time-sharing of the transition function requires very fast hardware. At 60 frames/s, the time-window allotted to each cell is approximately 160 ns. CAM's pipelined structure takes care of all the bookkeeping involved in locating a cell's neighbors, reading their values, buffering and eventually writing the cell's new value, managing the display, etc., so that the full width for the window is available to the circuitry that evaluates the transition function.

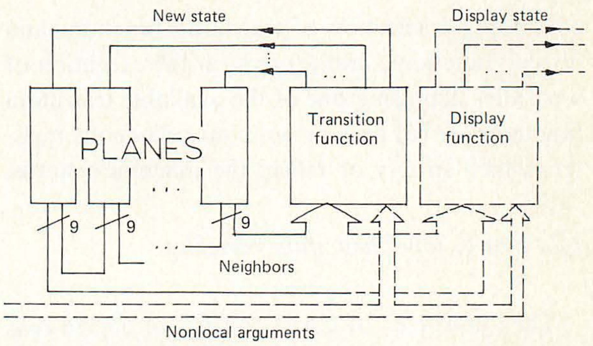


Fig. 1. CAM's basic computational loop (solid lines), with nonlocal arguments and display function (dashed lines).

3. Architectural notes

3.1. *"Hard" vs. "soft" programming*

In CAM, the basic computational loop has the form of fig. 1. In principle, the functions in the above figure could be realized entirely by look-up tables; in this way, the behavior of the dynamical system under study could all be programmed in a "soft" way by merely downloading the tables with appropriate entries.

In practice, the totally soft approach is not viable. The number of binary lines that one might want to use – at one time or another – as *arguments* to the tables approaches one-hundred†, and the number of *result* bits (the width of a table entry) that one might want to use at one time or another is close to sixteen**. Thus, to achieve full programming generality in this context one would need a table of perhaps 16×10^{100} bits, which is clearly out of the question. On the other hand, such a large table would mostly do trivial *routing* of signals, which can be achieved with much less lavish means (cf. last footnote of this section).

In CAM, practical considerations (cost, bulk, power requirements, and timing) suggest a range of 2^{10}–2^{16} bytes as the maximum amount of fast RAM memory to be devoted to look-up tables‡.

In order best to use this resource, observe that only some of the lines available as potential function arguments – as well as only some of the potential result lines – are in practice used *at any one*

* This is achieved by a clever backtracking scheme that in effect instantly "flushes" the pipelines.

† Each of the eight planes supplies nine neighbors; sixteen more lines specify a cell's address; a few more lines may come from the random-number generators; and finally, additional lines may be used for specifying time-dependent behavior.

** One bit for each plane, to specify a cell's new state, plus approximately as many bits to specify the value of the corresponding pixel.

‡ Note that look-up tables are most effectively utilized when there contents is nearly random, and certainly few users would want to deliberately and independently assign millions of bits in order to specify the dynamics of a system.

time. Moreover, especially when very many arguments are actually used (e.g., many states per cell *and* many neighbors), transition functions of practical interest tend to display some regularity; because of this, with some simple preprocessing of the arguments or postprocessing of the results one can often replace an extremely large table by one of manageable size*.

For the above reasons, in CAM the tables' input and output lines are not permanently wired to the rest of the circuit (planes, display, etc.), but are made accessible to the user, through jumpers on the backplane. While this arrangement gives CAM enormous flexibility, without further support it might entail frequent manual interventions on the user's part. Yet, it is desirable to run long unattended series of experiments or complex live demos totally under software control. This problem is solved by providing a number of tables, rather than a single one. Each table can be wired independently of the others, and thus different combinations of sources and destination lines are available at the same time. Which combination (or mix of combinations) is actually used for a given step – in other words, which of the many available rules should be used at any moment – is determined by a *channel* parameter which accompanies the request to execute a step. An additional parameter, called the *phase*, may be supplied alongside with the channel, to further characterize the selected rule.

Channels and phases (further discussed in the following section) provide an elegant compromise to the "hard-vs-soft programming" dilemma†.

* This solution can actually take many different forms: one can cascade two or more tables, use a hybrid of tables and hardwired circuitry, or even "microprogram" a complex functions as a sequence of several cellular-automaton steps (cf. section 5.3).

† Of course, another solution is to replace jumpers by a software-controlled "electronic switchboard" having access to all signal points of potential interest. The natural place for such a router card, which the user should provide himself, is in a slot between the planes cards and the tables cards (cf. section 4.).

3.2. *Channel and phase*

In the previous section, we discussed the loop of fig. 1 as viewed from the functions' side. Here, we shall discuss it from the planes' side.

During execution of a step, the memory planes (which, stacked together, make up the state of the cellular automaton) are scanned concurrently, cell by cell, and the contents of each plane is cycled through a complex pipeline arrangement whose details need not concern us. All we need know is that at one point in the pipeline there is an intentional gap, which must be filled by the user in order to close the loop. This gap has the form of

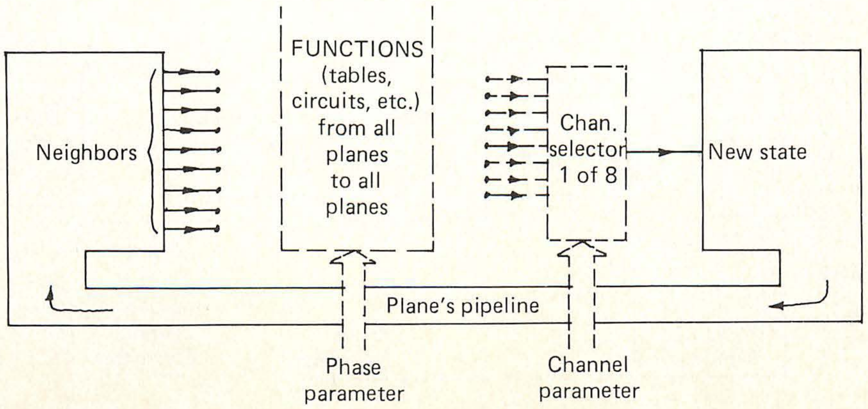


Fig. 2. CAM's pipeline (solid lines), with gap to be filled by the transition function. In the dashed portion, the channel selector connects the pipeline input to any one of 8 different sources, and the phase parameter can be used to modify the behavior of such sources. Any interactions between pipelines takes place in the "fuctions" box, where all of the random wiring is located.

fig. 2: on the pipeline there are nine output taps, which supply the states of the cell's nine neighbors, and one input tap, which is to receive the cell's new state*.

It is across the gap – and only there – that the pipelines may interact with one other and with the "external" world (nonlocal arguments, time-dependent parameters, etc.), since the user may inject anything he wishes in the input tap.

In CAM, the pipeline's input tap is preceded by an 8-line data selector (fig. 2) controlled by the channel parameter. In this way, what is fed into the input tap can be made to come, under software control, from any one of eight predefined sources (any combination of wires, tables, and discrete circuits). The same channel parameter applies to all planes, so that the selections made by the planes are coordinated. In addition, the phase parameter is made available to those sources, to be decoded at will. Briefly, at each step the channel *selects* a source for each plane and the phase *parametrizes* it..

We shall give a specific example of the above ideas. In another paper in these proceedings [2], Margolus describes a very clever cellular-automaton realization of the billiard-ball model of computation. Margolus's rule uses as a neighborhood only a two-by-two square contained in the standard nine-cell neighborhood; however, which of the four such squares is used by a given cell at a given time depends on the parity of time and on the parity of the horizontal and vertical coordinates. Thus the rule is space- and time-dependent (though in a very simple way), and requires a look-up table with 12 address lines (9 for the neighbors, 2 for horizontal and vertical parity, and 1 for time parity) or $2^{12}(=4096)$ bits. Other interesting rules can be defined on this "Margolus neighborhood;" in order to have several of these rules available in the machine at the same time (for instance, for comparing their behavior without having to reload the rule every time), as many 4096-bit RAM tables would be required. But the extraction of the four relevant neighbors out of the nine available can be done once and for all by a simple circuit or a look-up table (fig. 3 uses a combination of both), thus reducing the size of the table needed for each rule from 4096 bits to only $2^4(=16)$ bits! The wiring for this situation is illustrated in fig. 3. Note that the bit indicating time parity is supplied through the phase parameter, and thus by the software, rather than by a hardware counter. This is desirable since in Margolus's rule a reversal of the time phase entails

*Since we are now dealing with just one plane, by "state" of a cell we mean the one-bit state component of that cell in that particular plane.

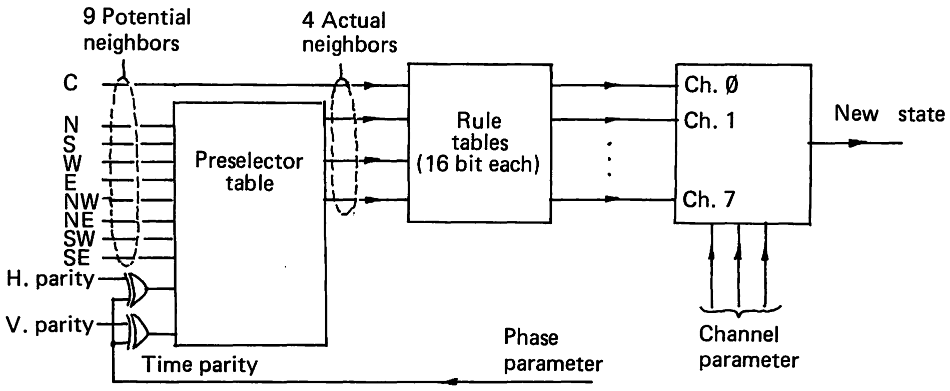


Fig. 3. Portion of CAM wired to run several rules based on the Margolus neighborhood. Neighbor "preselection" is done by a combination of hardware and table look-up, so as to permit use of smaller tables for the rules themselves. The phase bit determines whether a rule should execute forward or time-reversed.

running the rule backwards in time; thus the billiard-ball computation can be made to go forward or backward in time under program control. In this example, then, the *channel* selects one of several rules sharing the Margolus neighborhood, and the *phase* determines the direction of time.

3.3. *Data structures**

CAM's architecture provides logical space for the following data *objects*:

REGISTERS:
8 *system registers*
8 *user registers*
AREAS:
8 cellular-automaton *planes*
8 function *tables*

System and user registers are all one byte wide†. The planes consist of 8K bytes each (256 × 256 = 64K bits = 3K bytes), and the tables may be of arbitrary size (typically from 1K to 8K bytes).

User access to the objects is through the user port, whose lines are directly buffered into an asynchronous bus structured as follows:

2 lines for selecting among the four major groups of objects (system registers, user registers, planes, and tables),

3 lines for selecting one of the eight objects in each class,

1 line for data direction (read/write),

8 bidirectional data lines, and

1 data strobe line.

The final selection of an individual byte within an area (plane or a table) is accomplished by the *cursor*, an internal 16-bit counter of which 14 bits are bussed. The cursor may be set by the user to point at any place within an area, and normally autoincrements after each strobe pulse (the architecture reserves a *cursor control* user register for more sophisticated uses of the cursor).

3.4. *Scanning and stepping*

CAM continually generates sync signals for the display in a cycle called *frame* that last 1/60 s.

Normally, during each frame CAM also *scans* the whole array (addressing bytes through the *scanner*, an internal 13-bit counter) and displays its contents; if a *step* is executed during that scan (see below), then the old state of the planes is replaced by the new state, otherwise it remains unchanged. On the other hand, if at the beginning of the frame the bus is found in a state which requests input/output from any plane, then scanning is suppressed for that frame and access to the array is granted to the bus, which will address it through the cursor; in this case the frame is blanked.

In order to schedule a step, the user writes to system register 0 (the *step* register) a byte containing the desired channel and phase. The step will be executed on the next available scan frame.

Two output lines ("step pending" and "scan in progress") in the user port supply all synchronization information needed by the driving software.

4. Hardware and software

The following is a description of the hardware and software that currently comprise the CAM package.

* In this architectural overview, as a rule we state a definite *base* value for the maximum number or size of certain resources, such as the number of planes, the number of sources selectable by the channel parameter, the range of the areas cursor, etc. Only this base value is fully supported by CAM's hardware (in terms of fan-cut capabilities, number of address lines, slots in the card cage, power-supply requirements, etc.). However, the system's architecture allows unlimited expansion beyond the base value for most of these parameters, and explicit "hooks" are provided for making the expansion as straightforward as possible.

† System registers are either actually used in the current version (e.g., stpe register, cursor low byte, cursor high byte, utility latch register, etc.) or their address space is reserved for future versions of CAM. On the other hand, the implementation and use of user registers is left to the discretion of the user.

4.1. *Hardware*

The hardware of CAM consists of a set of 4.5" × 6.5" cards in a 10-slot card cage with power supply. Half of the backplane is occupied by the system bus; the other half consists of discrete input or output pins which can be wire-wrapped or jumpered by the user (cf. section 3.1) to custom-configure the system.

In its basic configuration, CAM is "populated" with just four cards, namely:

(a) The *interface* card, containing the user port, the monitor port, system registers, the cursor counter, and other interfacing logic. A simple random-number generator is usually plugged into this card.

(b) Tne *scanner* card, containing all the timing, control, and address circuitry involved in parsing a frame, scanning the planes, and operating the pipelines.

(c) One *planes* card, containing two planes with the associated memory, pipelines, buffers, and multiplexers.

(d) One *tables* card, containing two separately addressable 4 × 1024-bit tables, with their access circuitry.

Additional planes and tables cards may be added to this basic configuration, in order to have more states per cell and a wider choice of rules.

CAM's backplane has been structured keeping in mind that users may want to plug in additional custom-designed cards of their own (e.g., more sophisticated random-number generators, hard-wired or PROM functions, prototyping boards with random logic, etc.).

4.2. *Interfacing*

The monitor port supplies 75-ohm signals for Red, Blue, Green, Sync, and Black-and-White Composite. Composite Sync is also superposed on Red, Blue, and Green, for convenience, but may be disabled. Separate horizontal and vertical syncs are available on the backplane.

The user port can be driven by a standard PIA (parallel interface adaptor chip) such as the MC6821, or by any parallel port providing 16 bi-directional lines. In particular, the required lines from an internal PIA are available at the jacks of an ATARI 400/800 personal computer.

4.3. *Software*

The low-level software required to drive CAM is extremely simple (and, in fact, CAM can be easily hand-tested by driving the user port with a few switches). Each atomic operation involves addressing a particular data object (cf. section 3.3), and transferring a byte to or from it. Owing to the autoincrement feature of CAM's cursor and the autostrobe feature of the PIA, an entire plane or table can be downloaded by repeatedly writing to the same PIA data register. And once planes and tables are initialized, the cellular automaton is run by selecting the step register and repeatedly writing the desired channel-and-phase byte to it.

CAM's hardware is accompanied by an installation guide describing in machine-independent form the basic control subroutines.

At the moment, we are controlling CAM by means of a sophisticated software package which runsd on the ATARI 400/800 personal computer*. This package assists the user in creating, filing, and loading function tables and plane configurations, and in running the cellular automaton (single-step, variable speed, step through a cycle of different funtions, etc.). A *window* system allows the user to view with magnification on the ATARI's screen a selected portion of the automaton, and to manipulate this area with ease.

We hope that, owing to the fact that CAM constitutes a *de facto* standard, its users will be able to share data, applications, and software developed for other host computers.

* This package was created by Norman Margolus, who also contributed in a substantial way to the functional definition of CAM and developed a number of origina applications. Interesting application and much user feedback were provided also by Gérard Vichniac.

5. Extensions

Though the range of applications that can be explored with CAM "as given" is enormous, some applications may demand features (such as array size and wrap-around, number of neighbors, etc.) different from those hardwired in CAM. Here, we shall briefly indicate how one cay synthesize a much wider range of features. In certain cases this will require additional software or hardware, and possibly entail a speed penalty; what is important is that one can at least "preview" the behavior of very peculiar systems without having to build from scratch a special-purpose machine.

5.1. *Neighbors*

Certain transition functions may require a neighborhood template in which one or more neighbors lie outside of the 3 × 3 "window" provided by the CAM hardware. The extra neighbors can be made to appear through the 3 × 3 window by putting on auxiliary planes suitably shifted copies of the original planes. Thus, one would run a two-phase cycle: one phase performs trivial *shift* steps from the original planes to the auxiliary ones, and the other performs the desired transition-function step. Note that a built-in hardware feature allows the user to blank out intermediate results*.

5.2. *Array size*

The evolution of an arbitrarily large array can be simulated on CAM with little overhead. The whole array is stored in the host computer's memory, and is partitioned into blocks of size somewhat smaller that 256 × 256. In turn each block is loaded into the CAM array, surrounded by an edge coming from the adjacent blocks, and run for a few steps, until the edges (which are cut off from the rest of the array) are fully degraded and only the block itself contains correct information. The host's memory is then updated accordingly, and the process is repeated.

If the target array's size is not too large a multiple of 256 × 256, another technique is possible which does not impose any speed penalty. Intuitively, several tori (i.e., wrapped-around planes) be cut open, unfolded, and then "sewn together" edge-to-edge so as to make up a larger torus. This can be done with some external circuitry which decodes the address of the edges and makes one plane look for edge neighbors for another plane.

5.3. *Boundary conditions, sources and sinks, "miracles"*

As mentioned above and in section 2.3, the wrap-around of the planes can be disabled and replaced by other boundary conditions by decoding the address of the edges, or by means of auxiliary planes loaded with suitable edge patterns.

These techniques can also be used whenever one wants to force a part of the system to obey laws that are not an implicit consequence of the given local rule. For instance, the cellular automaton's state-variables may be thought of as representing the relatively few mechanical modes (degrees of freedom) of a system; then the average effect of a much larger number of thermal modes can be simulated by couplig the system with thermal sources (e.g., a random-number generator) and sinks (e.g., an absorbing wall).

Finally, the evolution of a system can be steered towards rare and more interesting events by "miracles" performed by the user. Specific objects can be moved, created, or zapped out of existence; a sequence of events can be traced in slow motion, analyzed, and modified, much as in tracing a computer program; and – if the transition function is invertible – the direction of time itself can be reversed.

* At 60 frames per second, a display that comes on every other frame (or even every third or fourth frame) still gives a reasonably continuous view of a system's dynamics.

5.4. *Externally-driven parallel processing*

Certain parallel processing applications, while still substantially adhering to the *locality* and *uniformity* of the cellular-automaton paradigm, strongly depart from *time-independence*. Rather, the system's dynamics is "driven" by an external sequencer through a fixed or variable sequence of different dynamical rules. Typical examples are certain forms of image processing (e.g., chromosome identification and counting) or the simulation of microcoded VLSI arrays. In this case, instead of preloading a large number of look-up tables with all the possible rules prescribed by the sequence, it may be more convenient to dynamically download tables as they are needed. This can always be done between steps, at the cost of slowing down the system. However, in CAM tables can also be written "on-the-fly", that is, during execution of a step. Thus, if one wants to retain maximum speed, one can use two sets of tables to be used in alternation; at any moment, the one which is not in use can be loaded concurrently with the ongoing step.

6. Conclusions

CAM is a powerful tool for experiments in parallel dynamics. It can be configured for a number of practical applications involving parallel computation and image processing. Its great flexibility and moderate cost make it likely that it will be used by a large number of groups, and that there will be a lively exchange of applications between groups.

The real-time graphic display and the essentially interactive nature of the machine are invaluable for research purposes, and also provide demos and "living slides" having great visual impact and didactic potential.

References

[1] A. Hoogland et al., "A Special-Purpose Processor for the Monte-Carlo Simulation of Ising Spin Systems," preprint, Laboratory of Applied Physics, University of Technology, Delft, the Netherlands.

[2] Norman Margolus, "Physics-like Models of Computation," Physica 10D (1984) 81 (these proceedings).

[3] Robert B. Pearson, John Richardson and Doug Toussaint, "A Special-Purpose Machine for Monte-Carlo simulation," preprint, Institute for Theoretical Physics, University of California, Santa Barbara (1981).

[4] Tommaso Toffoli, "Cellular Automata Mechanics," Tech. Rep. No. 208, Logic of Computers Group, CCS Dept., The University of Michigan (November 1977).

[5] Tommaso Toffoli, "Cellular Automata as an Alternative to (rather than an approximation of) Differential Equations in Modeling Physics, Physica 10D (1984) 117 (these proceedings).

[6] Gérard Vichniac, "Simulating Physics with Cellular Automata", Physica 10D (1984) 96 (these proceedings).

1026556669

Phenomenology of Two-Dimensional Cellular Automata*

Norman H. Packard and Stephen Wolfram

The Institute for Advanced Study, Princeton NJ 08540.

May 1984

ABSTRACT

A largely empirical study of two-dimensional cellular automata is presented. Classes of behaviour similar to those in one-dimensional cellular automata are found. Several geometrical phenomena, including dendritic growth and labyrinthine domain formation, are discussed. The undecidability of some global properties of two-dimensional cellular automata is mentioned.

* Work supported in part by the U.S. Office of Naval Research under contract number N00014-80-C-0657.

1. Introduction

Cellular automata are mathematical models for systems in which many simple components act together to produce complicated patterns of behaviour. One-dimensional cellular automata have now been investigated in several ways ([1], and references therein). This paper presents an exploratory study of two-dimensional cellular automata*. The extension to two dimensions is significant for comparisons with many experimental results on pattern formation in physical systems. Immediate applications include dendritic crystal growth [6], reaction-diffusion systems, and turbulent flow patterns. (The Navier-Stokes equations for fluid flow appear to admit turbulent solutions only in two or more dimensions.)

A cellular automaton consists of a regular lattice of sites. Each site takes on k possible values, and is updated in discrete time steps according to a rule ϕ that depends on the values of sites in some neighbourhood around it. The value a_i of a site at position i in a one-dimensional cellular automata with a rule that oniy nearest neighbours thus evolves according to

$$a_i^{(t+1)} = \phi[a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)}] \ . \tag{1.1}$$

There are several possible lattices and neighbourhood structures for two-dimensional cellular automata. This paper considers primarily square and hexagonal lattices, with the three neighbourhood structures illustrated in figure 1.1. A five-neighbour square cellular automaton then evolves in analogy with eqn. (1.1) according to

$$a_{i,j}^{(t+1)} = \phi[a_{i,j}^{(t)}, a_{i,j+1}^{(t)}, a_{i+1,j}^{(t)}, a_{i,j-1}^{(t)}, a_{i-1,j}^{(t)}] \ . \tag{1.2}$$

General cellular automaton rules may be labelled as indicated in figure 1.1. Here we usually consider the special class of totalistic rules, in which the value of a site depends only on the sum of the values in the neighbourhood:

$$a_{i,j}^{(t+1)} = f\ [a_{i,j}^{(t)} + a_{i,j+1}^{(t)} + a_{i+1,j}^{(t)} + a_{i,j-1}^{(t)} + a_{i-1,j}^{(t)}] \ . \tag{1.3}$$

These rules are conveniently specified by a code [7]

$$\mathbf{C} = \sum_n f\,(n)k^n \ . \tag{1.4}$$

We also consider outer totalistic rules, in which the value of a site depends separately on the sum of the values of sites in a neighbourhood, and on the previous value of the site itself:

$$a_{i,j}^{(t+1)} = \tilde{f}\ (a_{i,j}^{(t)}, a_{i,j+1}^{(t)} + a_{i+1,j}^{(t)} + a_{i,j-1}^{(t)} + a_{i-1,j}^{(t)}) \ . \tag{1.5}$$

Such rules are specified by a code

$$\tilde{\mathbf{C}} = \sum_n \tilde{f}\ [a,n]k^{kn+a} \ . \tag{1.6}$$

This paper considers two-dimensional cellular automata with two possible values at each site, $k=2$. Table 1 gives the number of possible rules of various kinds for such cellular automata. A notorious example of an outer totalistic nine-neighbour square cellular automaton is the "Game of Life" [8], with a rule specified by code $\tilde{\mathbf{C}}=224$.

Despite the simplicity of their construction, cellular automata are found to be capable of very complicated behaviour. Direct mathematical analysis is in general of little utility in elucidating their properties. One must at first resort to empirical means. This paper is a phenomenological study of typical two-dimensional cellular automata. Its approach is largely experimental in character: cellular automaton rules are selected and their evolution from various initial states is traced by explicit simulation**. The emphasis is on generic properties. Typical initial states are

* Some aspects of two-dimensional cellular automata were discussed in [2,3], and mentioned in [4]. Additive two-dimensional cellular automata were considered in [5].

** Two computer systems were used. The first was the special-purpose pipelined TTL machine built by the M.I.T. Digital Information Mechanics group [9]. This machine updates all sites on a 256×256 square cellular automaton lattice 60 times per second. It is controlled by a microcomputer, with software written in FORTH. It allows for five- and nine-neighbour rules, with up to four effective values for each site. The second system was chosen. Except for some restricted kinds of rules, table 1 shows that the number of possible cellular automaton rules is far too great for each to be investigated explicitly. For the most part one must resort to random sampling, with the expectation that the rules so selected are typical. The phenomena identified by this experimental approach may then be investigated in detail using analytical approximations, and by conventional mathematical means. Generic properties are significant because they are independent of precise details of cellular automaton construction, and may be expected to be universal to a wide class of systems, including those that occur in nature.

Empirical studies strongly suggest that the qualitative properties of one-dimensional cellular automata are largely independent of such features of their construction as the number of possible values for each site, and the size of the neighbourhood. Four qualitative classes of behaviour have been identified in one-dimensional cellular automata [7]. Starting from typical initial configurations, class 1 cellular automata evolve to homogeneous final states. Class 2 cellular automata yield separated periodic structures. Class 3 cellular automata exhibit chaotic behaviour, and yield aperiodic patterns. Small changes in initial states usually lead to linearly-increasing regions of change. Class 4 cellular automata exhibit complicated localized and propagating structures. Cellular automata may be considered as information-processing systems, their evolution performing some computation on the sequence of site values given as the initial state. It is conjectured that class 4 cellular automata are generically capable of universal computation, so that they can implement arbitrary information-processing procedures.

Dynamical systems theory methods may be used to investigate the global properties of cellular automata. One considers the set of configurations generated after some time from any possible initial configuration. Most cellular automaton mappings are irreversible (and not surjective), so that the set of configurations generated contracts with time. Class 1 cellular automata evolve from almost all initial states to a unique final state, analogous to a fixed point. Class 2 cellular automata evolve to collections of periodic structures, analogous to limit cycles. The contraction of the set of configurations generated by a cellular automaton is reflected in a decrease in its entropy or dimension. Starting from all possible initial configurations (corresponding to a set defined to have dimension one), class 3 cellular automata yield sets of configurations with smaller, but positive, dimensions. These sets are directly analogous to the chaotic (or "strange") attractors found in some continuous dynamical systems (e.g. [10]).

Entropy or dimension gives only a coarse characterization of sets of cellular automaton configurations. Formal language theory (e.g. [11]) provides a more complete and detailed characterization [12]. Configurations may be considered as words in a formal language; sets of configurations are specified by the grammatical rules of the language. The set of configurations generated after any finite number of time steps in the evolution of a one-dimensional cellular automaton can be shown to form a regular language: the possible configurations thus correspond to possible paths through a finite graph. For most class 3 and 4 cellular automata, the complexity of this graph grows rapidly with time, so that the limit set is presumably not a regular language (cf. [13]).

This paper reports evidence that the global properties of two-dimensional cellular automata are very similar to those one-dimensional cellular automata. Many of the local phenomena found in two-dimensional cellular automata also have analogues in one dimension. However, there are a variety of phenomena that depend on the geometry of the two-dimensional lattice. Many of these phenomena involve complicated boundaries and interfaces, which have no direct analogue in one dimension.

Section 2 discusses the evolution of two-dimensional cellular automata from simple "seeds", consisting of a few nonzero initial sites. Just as in one dimension, some cellular automata give

---

a software program running on the Ridge 32 computer. The kernel was written in assembly language; the top-level interface in C. (Requests for copies of this program should be directed to the authors.) A 128×128 cellular automaton lattice is typically updated about 7 times per second. One-dimensional cellular automaton simulations were carried out with our CA cellular automaton simulation package, written in the C programming language, usually running on a Sun Workstation.

regular and self-similar patterns; others yield complicated and apparently random patterns. A new feature in two dimensions is the generation of patterns with dendritic boundaries, much as observed in many natural systems. Most two-dimensional patterns generated by cellular automaton growth have a polytopic boundary that reflects the structure of the neighbourhood in the cellular automaton rule (cf. [14]). Some rules, however, yield slowly-growing patterns that tend to a circular shape independent of the underlying cellular automaton lattice.

Section 3 considers evolution from typical disordered initial states. Some cellular automata evolve to stationary structures analogous to crystalline forms. The boundaries between domains of different phases may behave as if they carry a surface tension: positive surface tensions lead to large smooth-walled domains; negative surface tensions give rise to labyrinthine structures with highly-convoluted walls. Other cellular automata yield chaotic, class 3, behaviour. Small changes in their initial configurations lead to linearly-increasing regions of change, usually circular or at least rounded.

Section 4 discusses some quantitative characterizations of the global properties of two-dimensional cellular automata. Many definitions are carried through directly from one dimension, but some results are rather different. In particular, the sets generated after a finite number of time steps of cellular automaton evolution may no longer be described by regular languages, but are in fact in general non-recursive. As a consequence, several global properties that are decidable for one-dimensional cellular automata become undecidable in two dimensions (cf. [15]).

2. Evolution from simple seeds

This section discusses patterns formed by the evolution of cellular automata from simple seeds. The seeds consist of single nonzero sites, or small regions containing a few nonzero sites, in a background of zero sites. The growth of cellular automata from such initial conditions should provide models for a variety of physical and other phenomena. One example is crystal growth [6]. The cellular automaton lattice follows the crystal lattice, with nonzero sites representing the presence of atoms or regions of the crystal. Different cellular automaton rules are found to yield both epitaxial (regular) and dendritic (snowflake-like) crystal structures. In other systems the seed may correspond to a small initial disturbance, which grows with time to produce a complicated structure. Such a phenomenon presumably occur when fluid turbulence develops downstream from an obstruction or orifice*.

Figure 2.1 shows some typical examples of patterns generated by the evolution of two-dimensional cellular automata from initial states containing a single nonzero site. In each case, the sequence of two-dimensional patterns formed is shown as a succession of "frames". A spacetime "section" is also shown, giving the evolution of the centre horizontal line in the two-dimensional lattice with time. Figure 2.2 gives some examples of analogous sections generated by typical one-dimensional cellular automata.

With some cellular automaton rules, simple seeds always die out, leaving the null configuration, in which all sites have value zero. With other rules, all or part of the initial seed may remain invariant with time, yielding a fixed pattern, independent of time. With many cellular automaton rules, however, a growing pattern is produced, as shown in figures 2.1 and 2.2.

Figures 2.1(a) and 2.2(a) illustrate the simple case in which a uniform "epitaxial" growing pattern is generated. At each time step a pattern with a fixed density of nonzero sites is produced. In the two-dimensional case, the pattern has a simple piecewise linear boundary. The set of nonzero sites form a pyramid of fixed density in spacetime (or a triangle in the one-dimensional case). The faces of this pyramid correspond to the surface traced out by the boundary of the growing pattern in time. Its edges lie along the directions of maximal growth.

Several dimensions may be defined to characterize the overall limiting structure of patterns generated by cellular automaton evolution. We give definitions for the case of cellular automata on a two-dimensional lattice; extensions to lattices with other dimensions are straightforward. Two general types of dimension may be defined. The first, denoted generically D, depend on the overall spacetime pattern. The second, denoted $\overline{D}$ depend only on the boundary of the pattern. In simple cases, the boundary may be defined as the set of sites which can be reached by some path from the outside without crossing any nonzero sites. In more complicated cases, where disconnected spatial patterns are produced, the definition may be modified to require that the path should no sites whose values have ever been nonzero on any previous time step. More complete definitions are exact only in the infinite time limit. One considers the intersection of the complete spacetime pattern, or of its boundary, with some hyperplane. Let the number of nonzero sites in this intersection be n. Then the corresponding dimension is given in terms of the asymptotic growth rate of n with some parameter t, usually time, as $D = \log_t n$. These dimensions are analogous to Kolmogorov dimensions, and are presumably in most cases equal to the corresponding Hausdorff dimensions (cf. [17]). In all cases, the maximum (minimum) possible value of D ($\overline{D}$) is the topological dimension of the hyperplane. Often the values D and $\overline{D}$ may be computed directly by geometrical methods.

The "spatial growth dimension" D_x is defined to be the dimension associated with spatial patterns generated at particular times t; $\overline{D}_x$ is defined to be the dimension associated with their boundaries. A complete definition of the spatial growth dimension requires a subtle large time limit to be taken.

The "total growth dimensions" D and $\overline{D}$ are taken to be the dimensions associated with the complete spacetime pattern, and with its boundary, respectively. These dimensions are usually

* A cellular automaton approximation to the Euler equations is given in [16].

evaluated by considering patterns formed up to some finite time T, and estimating their limit as $T \rightarrow \infty$.

The "temporal growth dimensions" $D_t^{(\vec{x})}$ and $\bar{D}_t^{(\vec{x})}$ are dimensions associated with sections through the spacetime pattern in spatial direction $\vec{x}$. The total growth dimension may evidently be obtained as an appropriate average of the temporal growth dimensions over possible directions. (The average must be taken over numbers of nonzero sites n, and so requires exponentiation of the growth dimensions.)

The maximal values for complete growth dimensions are given by the topological dimensions of the hyperplanes that define them. These topological dimensions give lower bounds on boundary growth dimensions. For the simple pattern of figure 2.1(a), all the growth dimensions achieve their extremal values: $D=3$, $\bar{D}=D_x=D_t=2$, $\bar{D}_x=\bar{D}_t=1$.

Figure 2.1(b) shows a pattern generated by a two-dimensional cellular automaton for which space-time sections in any direction exhibit an asymptotically self-similar or fractal form. Figure 2.2(b) shows a one-dimensional cellular automaton that yields section of the same form. The density of nonzero sites in these sections tends asymptotically to zero, showing that their temporal growth dimension is less than two. By a simple geometrical construction they are in fact found to have dimension $\log_2 3 \simeq 1.59$. For the cellular automaton of figure 2.1(b), therefore, $D_t^{(\vec{x})} = \log_2 3$, for any direction $\vec{x}$. The corresponding total dimension is simply $D = 1 + \log_2 3$. Although the interior of the pattern in figure 2.1(b) has a fractal form, its boundary is just as in figure 2.1(a). All boundary growth dimensions thus have their minimal values for figure 2.1(b).

The evaluation of the spatial growth dimension for figure 2.1(b) is slightly more complicated. It can be shown that the number of nonzero sites in the line of a section through the pattern of figure 2.1(b) corresponding to time t is given by $2^{\#_1(t)}$, where $\#_1(t)$ gives the number of nonzero digits in the binary representation of the integer t [4]. (to be continued...)

Figures 2.1(c) and 2.2(d) show examples of cellular automata which generate complicated patterns that are not homogeneous, but appear to have fixed nonzero asymptotic density. It is remarkable that simple rules, even starting from the simple initial conditions shown, can generate patterns of such apparent complexity. It seems likely that the iteration of the cellular automaton rule is essentially the simplest procedure by which these patterns can be specified. The pattern obtained after t time steps could then effectively be generated only by explicit evolution of the cellular automaton for time t. This is in contrast to the case of the patterns in figures 2.1(b), 2.2(b) and 2.2(c). The form of such self-similar patterns at any time t may be found by a simple fixed procedure or "closed form" expression. If the procedure were implemented as a recursive function, then the number of temporary storage elements would be fixed, independent of t. The function would thus be primitive recursive (e.g. [11]), and could be converted to a purely iterative form. On the other hand, for figures 2.1(c) and 2.2(d) it seems likely that no such simple fixed procedure can be given, and that a general recursive function is required. Such a function cannot be converted to an iterative form, and its evaluation in general requires a number of temporary storage elements that increases without bound with t.

The nonzero asymptotic density of the pattern in figure 2.2(c), and the regularity of its boundary imply that all of its growth dimensions are extremal just as for figure 2.1(a).

Figures 2.1(d), (e) and (f) illustrate cellular automata which give patterns that exhibit corrugated, dendritic, boundaries. Such complicated boundaries can have no analogue in one-dimensional cellular automata: they are a first example of a qualitative phenomenon that requires two or more dimensions.

Figure 2.1(d) is for the simple additive rule that takes the value of each site to be the sum modulo two of the previous values of all sites in its neighbourhood. The total growth dimension for this rule, and its analogues on d-dimensional lattices, is given by [4] $\log_2[d(\sqrt{1+4/d}+1)]$, or approximately 2.45 for $d=2$. The temporal growth dimension obtained by intersection with a spacetime hyperplane of topological dimension d' is given by the same formula, with d' replacing d.

The cellular automata of figures 2.1(e) and (f) give rise to patterns with nonzero asymptotic densities. Their complete growth dimensions are therefore maximal. Their boundaries are however corrugated. They have fractal forms analogous to Koch curves, and yield non-minimal growth dimensions. In each case, the pattern grows by producing "branches" along the four lattice directions. Each of these branches then in turn produces sidebranches, which themselves produce sidebranches, and so on. This recursive process yields a highly corrugated boundary. However, as the process continues, the sidebranches grow into each other, forming an essentially solid region. In fact, after each 2^j time steps the boundary takes on an essentially regular form. It is only between such times that a dendritic boundary is present.

Figure 2.1(e) is an example of a "solidification" cellular automaton rule [6], in which any site, once it attains value one, can never revert to value zero. Such rules are of significance in studies of processes such as crystal growth. Notice that although the interior of figure 2.1(e) takes on a fixed form with time, the possibility of a simple one-dimensional cellular automaton model for the boundary alone is precluded by its corrugated form.

Empirical studies indicate that among all (symmetric) two-dimensional cellular automata, patterns with the form of figure 2.1(c) are the most commonly generated. Fractal boundaries are comparatively common, but their growth dimensions are usually quite close to the minimal value. Fractal sections are also comparatively common for five-neighbour rules, but become less common for nine-neighbour rules.

Figure 2.3 shows the evolution of various two-dimensional cellular automata from initial states containing both single nonzero sites, and small regions with a few nonzero sites. After a long time, the overall patterns generated are seen to be largely independent of the particular form of the initial state. Deformations in boundaries occur only on length scales of order the size of the initial region of nonzero sites, and presumably become negligible in the infinite time limit. In cases such as (c) and (e), the form of the initial state leads to specific dislocations in the final pattern.

The rules for the all the two-dimensional cellular automata shown in figures 2.1 and 2.3 are completely invariant under all the rotation and reflection symmetry transformations on their neighbourhoods. Figure 2.4 shows patterns generated by cellular automaton rules with lower symmetries. These patterns are often complicated both in their boundaries and internal structure.

Cellular automaton rules embody a finite maximum information propagation speed. This implies the existence of a "bounding surface" expanding at this finite speed. All nonzero sites generated by cellular automaton evolution from a localized seed must lie within this bounding surface. (The cellular automata considered here leave a background of zero sites invariant; such a background must be mapped to itself after at most k time steps with any cellular automaton rule.)

The pattern generated after t time steps by any cellular automaton is always bounded by the polytope (planar-faced surface) corresponding to the "unit cell" formed from the set of vectors specifying the displacements of sites in the neighbourhood, magnified by a factor t in linear dimensions (cf. [14]). Thus patterns generated by five-neighbour cellular automaton rules always lie within an expanding diamond-shaped region, while those with nine-neighbour rules may fill out a square region.

The actual bounding surface for a particular cellular automaton rule often lies far inside the surface obtained by magnifying the unit cell. A sequence of better approximations to the bounding surface may be found as follows. First consider a set of sites representing the neighbourhood for a cellular automaton rule. If the centre site has value one at a particular time step, there could exist configurations for which all of the sites in the neighbourhood would attain value one on the next time step. However, there may be some sites whose values cannot change from zero to one in a single time step with any configuration. Growth does not occur along directions corresponding to such sites. The polytope formed from sites in the neighbourhood, excluding such sites, may be magnified by a factor t to yield a first approximation to the actual bounding surface for a cellular automaton rule. A better approximation is obtained from the polytope obtained after two time steps of cellular automaton evolution, magnified by a factor $t/2$.

The actual bounding surfaces for five-neighbour two-dimensional cellular automaton rules usually have their maximal diamond-shaped form. However, many nine-neighbour rules have a diamond-shaped form, rather than their maximal square form. Some nine-neighbour rules, such as those of figures 2.3(g) and (h) have octagonal bounding surfaces, while still others, such as those of figure 2.3(i) have dodecagonal bounding surfaces. The cellular automata rules with lower symmetries illustrated in figure 2.4 in many cases exhibit more complicated boundaries, with lower symmetries.

Patterns that maintain regular boundaries with time typically fill out their bounding surface at all times. Dendritic patterns, however, usually expand with the bounding surface only along a few axes. In other directions, they meet the bounding surface only at specific times, typically of the form 2^j. At other times, they lie within the bounding surface.

Dendritic boundaries are typically generated by cellular automaton rules that exhibit "growth inhibition". Growth inhibition occurs if there exist some a_i for which $\phi(a_1,..,0,..,a_n)=1$, but $\phi(a_1,..,0,..,a_n)$, or vice-versa. Such behaviour appears to be common in physical and other systems.

Figure 2.5 shows examples of two-dimensional cellular automata that exhibit the comparatively rare phenomenon of slow, diffusive, growth from simple seeds. Figure 2.6 gives a one-dimensional cellular automaton with essentially analogous behaviour.

The phenomenon is most easily discussed in the one-dimensional case. The pattern shown in figure 2.6 is such that it expands by one site at a particular time step only if the site on the boundary has value one. If the boundary site has one its other three possible nonzero values, then on average, no expansion occurs. The cellular automaton rule is such that the boundary sites have values one through four with roughly equal frequencies. Thus the pattern expands on average at a speed of about 1/4 sites per time step (on each side).

The origin of diffusive growth is similar in the two-dimensional case. Growth occurs there only when some particular several-site structure appears on the boundary. Many boundary structures occur with roughly equal probabilities, so that the average growth rate is small. A remarkable feature is that the boundaries of the patterns produced do not follow the polytopic form suggested by the underlying lattice construction of the cellular automaton. Instead, in many cases, asymptotically circular patterns appear to be produced.

3. Evolution from disordered initial states

In this section, we discuss the evolution of cellular automata from disordered initial states, in which each site is randomly chosen to have value zero or one (usually with probability). Such disordered configurations are typical members of the set of all possible configurations. Patterns generated from them are thus typical of those obtained with any initial state. The presence of structure in these patterns is an indication of self-organization in the cellular automaton.

As mentioned in section 1, four qualitative classes of behaviour have been identified in the evolution of one-dimensional cellular automata from disordered initial states. Examples of these classes are shown in figure 3.1. Figure 3.2 shows the evolution of some typical two-dimensional cellular automata from disordered initial states. The same four qualitative classes of behaviour may again be identified here. In fact, the space-time sections for two-dimensional cellular automata have a striking qualitative similarity to sections obtained from one-dimensional cellular automata, perhaps with some probabilistic noise added.

Just as in one dimension, some two-dimensional cellular automata evolve from almost all initial states to a unique homogeneous state, such as the null configuration. The final state for such class 1 cellular automata is usually reached after just a few time steps, but in some rare cases, there may be a long transient.

Figure 3.2(a) gives an example of a two-dimensional cellular automaton with class 2 behaviour. The disordered initial state evolves to a collection of separated simple structures, each stable or oscillatory with a small period. Each of these structures is a remnant of a particular feature in the initial state. The cellular automaton rule acts as a "filter" which preserves only certain features of the initial state. There is usually a simple pattern to the set of features preserved, and to the set of persistent structures produced. It should in fact be possible to devise cellular automaton rules that recognize particular sets of features, and to use such class 2 cellular automata for practical image processing tasks (cf. [18]).

The patterns generated by evolution from several different disordered configurations according to a particular cellular automaton rule are almost always qualitatively similar. Yet in many cases the cellular automaton evolution is unstable, in that small changes in the initial state lead to increasing changes in the patterns generated with time. Figures 3.1 and 3.2 include difference patterns that illustrate the effect of changing the value of a single site in the initial state. For class 2 cellular automata, such a change affects only a finite region, and the difference pattern remains bounded with time. Information propagates only a finite distance in class 2 cellular automata, so that a particular region of the final state is determined from a bounded region in the initial state. For class 3 cellular automata, on the other hand, information generically propagates at a nonzero speed forever, and a small change in the initial state affects an ever-increasing region. The difference patterns for class 3 cellular automata thus grow without bound, usually at a constant rate.

The locally periodic patterns generated after many time steps by class 2 cellular automata such as that in figure 3.2(a) consist of many separated structures located at essentially arbitrary positions. Figure 3.2(b) shows another form of class 2 cellular automaton. There are four basic "phases". Two have vertical stripes, with stripes either on even or odd sites. The other two phases have horizontal stripes. Regions that take on forms corresponding to one of these phases are invariant under the cellular automaton rule. Starting from a typical disordered state, each region in the cellular automaton lattice evolves towards a particular phase. At large times, the cellular automaton thus "crystallizes" into a patchwork of "domains". The domains consist of regions in particular phases. They are separated by domain walls. In the example of figure 3.2(b), these domain walls become essentially stationary after a finite time. A change in a single initial site produces a difference pattern that ultimately spreads only along the domain walls. The spread continues only so long as each successive region on the domain wall contains only particular arrangements of site values. The spread stops if a "pinning defect", corresponding to other arrangements of site values, is encountered. The arrangement of site values on the domain walls may in a first approximation be considered random. The difference pattern will thus spread forever only if the arrangements of site values necessary to support to support its propagation occur with a probability above the percolation threshold, so that they form an infinite connected cluster with probability one.

Figure 3.2(e) shows an example of a two-dimensional cellular automaton in which the domain walls can continue to move forever, essentially by a diffusion process. Figure 3.1(d) shows a one-dimensional cellular automaton with domain walls that exhibit analogous behaviour (cf. [19,20]). In both cases, some domains become progressively larger with time, while others eventually disappear completely. The domain walls in figure 3.2(e) behave as if they carry a positive surface tension; the diffusion process responsible for their movement is biased to reduce the local curvature of the interface. At large times, therefore, the domains either shrink to zero size, or have walls with continually decreasing curvature.

Figure 3.2(c) shows a two-dimensional cellular automaton with structures analogous to domains walls that carry a negative surface tension. More and more convoluted patterns are obtained with time. The resulting labyrinthine state is strongly reminiscent of behaviour observed with ferrofluids or magnetic bubbles [21].

Figures 3.2(f), (g) and (h) are examples of two-dimensional cellular automata that exhibit class 3 behaviour. Chaotic aperiodic patterns are obtained at all times. Moreover, the difference patterns resulting from changes in single initial site values expand at a fixed rate forever. A remarkable feature is that in almost all cases (figure 3.2(h) is an exception), the expansion occurs at the same speed in all directions, resulting in an asymptotically circular difference pattern. For some rules, the expansion occurs at maximal speed; but often the speed is about 0.8 times the maximum. When the difference patterns are not exactly circular, they tend to have rounded corner. And even with asymmetrical rules, circular difference patterns are often obtained. A rough analogue of this behaviour is found in asymmetric one-dimensional cellular automata which generate symmetrical difference patterns. Such behaviour is found to become increasingly common as k and r increase, or as the number of independent parameters in the rule ϕ increases.

An argument based on the central limit theorem suggests an explanation for the appearance of circular difference patterns in two-dimensional class 3 cellular automata. Consider the set of sites corresponding to the neighbourhood for a cellular automaton rule. For each site, compute the probability that the value of that site changes after one time step of cellular automaton evolution when the value of the centre site is changed, averaged over all possible arrangements of site values in the neighbourhood. An approximation to the probability distribution of differences is then obtained as a multiple convolution of this kernel. The number of convolutions performed increases with time. If the number of neighbourhood arrangements is sufficiently large, the kernel tends to be quite smooth. Convolutions of the kernel thus tend to a Gaussian form, independent of direction.

Some asymmetric class 3 cellular automata yield difference patterns that expand say in the horizontal direction, but contract in the vertical direction. At large times, such cellular automata produce patterns consisting of many independent horizontal lines, each behaving essentially as a one-dimensional class 3 cellular automaton.

Class 3 behaviour is considerably the commonest among two-dimensional cellular automata, just as it is for one-dimensional cellular automata with large k and r. It appears that as the number of parameters or degrees of freedom in a cellular automaton rule increases, there is a higher probability for some degree of freedom to show chaotic behaviour, leading to overall chaotic behaviour.

Figures 3.1(i) show examples of class 4 one-dimensional cellular automata. A characteristic feature of class 4 cellular automata is the existence of a complicated set of persistent structures, some of which propagate with time. Class 4 rules appear to occur with a frequency of a few percent among all one-dimensional cellular automaton rules. Often one suspects that some degrees of freedom in a cellular automaton exhibit class 4 behaviour, but they are masked by overall chaotic class 3 behaviour.

Class 4 cellular automata appear to be much less common in two dimensions than in one dimension. Figure 3.2(i) shows the evolution of a two-dimensional cellular automaton known as the "Game of Life" [8]. Many persistent structures, some propagating, have been identified in this cellular automaton. It has in addition been shown that these structures can be combined to perform arbitrary information processing, so that the cellular automaton supports universal computation [8]. Starting from a disordered initial state, the density of propagating structures ("gliders") produced is about one per 2000 site region.

Except for a few simple variants on the Game of Life, no other definite class 4 two-dimensional cellular automata were found in a random sample of several thousand outer totalistic rules*. of Some rules that appeared to be of class 2 were found to have long transients, characteristic of class 4 behaviour, but no propagating structures were seen. Other rules seemed to exhibit some class 4 features, but they were overwhelmed by dominant class 3 behaviour.

* A few examples of class 4 behaviour were however found among general rules. Requests for copies of the relevant rule tables should be directed to the authors.

4. Global properties

Section 3 discussed the evolution of cellular automata from particular typical initial states. This section considers the global properties of cellular automata, determined by evolution from all possible initial states.

Since most cellular automaton rules are irreversible, the set of configurations generated in evolution from all possible initial configurations typically contracts with time. Different classes of cellular automata yield limiting sets at large times with different structures.

Entropy and dimension provide quantitative characterizations of the sizes of sets of cellular automaton configurations [7]. The spatial set entropy or dimension for a set of two-dimensional cellular automaton configurations is defined by considering a $X_1 \times X_2$ patch of sites. If the set contains all possible configurations, then all $k^{X_1 X_2}$ different arrangements of site values must occur in the patch. In general let $N(X_1, X_2)$ different arrangements occur in the patch. Then the set entropy or dimension is defined as the limit of $1/(X_1 X_2) \log_k N(X_1, X_2)$ as $X_i \to \infty$. If the cellular automaton mapping is surjective, so that all possible configurations occur, then this dimension is equal to one. In general it decreases through irreversible cellular automaton evolution.

The spatial entropy for configurations generated by, say, one time step of cellular automaton evolution could be calculated by identifying the set of configurations that have predecessors under the cellular automaton mapping. In a one-dimensional cellular automaton, a block of length X is determined by a block of length $X+2r$ on the previous time step. A block of any length X can thus have at most $k^{X+2r}/k^X = k^{2r}$ predecessors (each of length $X+2r$). There is thus a fixed finite procedure to determine whether any given block can occur [22]. As a consequence, a finite characterization of the set of all blocks that occur may be given [12]. Each block corresponds to a word in a regular language. The regular language is specified by a finite graph; each word corresponds to a particular path through the graph. The graph has at most $2^{k^{2r}}$ nodes, each node representing roughly a possible subset of the set of k^{2r} predecessors.

In a two-dimensional cellular automaton with nearest-neighbour rules, a $X_1 \times X_2$ patch of sites is determined by a $(X_1+2)(X_2+2)$ patch on the previous time step. The number of possible predecessors thus grows with the perimeter of the patch, as $k^{2(X_1+X_2+2)}$. There is thus in general no bounded procedure to determine whether patches of any size can be generated in the evolution of a two-dimensional cellular automaton. As a consequence, there are strong indications that the set of configurations obtained after a finite number of time steps of two-dimensional cellular automaton evolution is not recursive.

Some cellular automaton rules lead to surjective mappings which allow all possible configurations to be generated at any time. There is a finite procedure to determine whether a particular one-dimensional cellular automaton rule leads to a surjective mapping. It appears however that there can be no analogous general finite procedure for two-dimensional cellular automata: the problem of determining surjectivity is undecidable in this case [15].

The set of configurations with a particular period under a one-dimensional cellular automaton mapping forms a finite complement language (subshift of finite type), in which only a fixed set of finite blocks of site values are excluded [12]. Periodic configurations for a two-dimensional cellular automaton are equivalent to tilings of a plane with dominoes so that all their edge colours match. The problem of finding such tilings in general undecidable [23]. Hence it is presumably undecidable whether any configurations of given period exist in a particular two-dimensional cellular automaton. If they do exist, then they may in general form form a non-recursive set.

In addition to considering the set of configurations generated at a particular time step in the evolution of a cellular automaton, one may also discuss sequences of site values obtained with time. In general, one may define entropies or dimensions by counting the numbers of different arrangements of site values that occur in regions on various hyperplanes through the spacetime pattern formed by the cellular automaton evolution. The magnitude of these entropies and dimensions is largely determined by the degree of correlation between site values in different parts of the regions. Two site values may be correlated only if information can travel from one to the other.

The difference patterns discussed in section 3 provide measures of information propagation. The rate of expansion of these difference patterns in some direction may be considered as the Lyapunov exponent in that direction for the cellular automaton evolution [7,24]. One may then derive a family of inequalities between the Lyapunov exponents and dimensions or entropies for two-dimensional cellular automata, largely analogous to those for one-dimensional cellular automata [7]. One notable phenomenon is that the invariant entropy of a two-dimensional cellular automaton mapping, obtained by considering the number of arrangements of site values in a spacetime tube long in the time direction (normalized by the total volume of the tube), vanishes unless the Lyapunov exponent is positive in all directions.

Acknowledgements

We are grateful to the M.I.T. Digital Information Mechanics group (E. Fredkin, N. Margolus, T. Toffoli and G. Vichniac) for the use of their special-purpose two-dimensional cellular automaton simulation machine, and for their hospitality and assistance. We thank R. Shaw for writing the kernel of our software simulation system for two-dimensional cellular automata. We are grateful to those mentioned and to C. Bennett, J. Crutchfield, L. Hurd and J. Milnor for discussions.

References

1. S. Wolfram, "Cellular automata: towards a paradigm for complexity", to be published in Nature.
2. T. Toffoli, "Cellular automata mechanics", PhD thesis, Logic of Computers Group, University of Michigan (1977).
3. G. Vichniac, "Simulating physics with cellular automata", Physica 10D (1984) 96.
4. S. Wolfram, "Statistical mechanics of cellular automata", Rev. Mod. Phys. 55 (1983) 601.
5. O. Martin, A. Odlyzko and S. Wolfram, "Algebraic properties of cellular automata", Comm. Math. Phys., (1984).
6. N. Packard, "Cellular automaton models for dendritic crystal growth", Institute for Advanced Study preprint (1984).
7. S. Wolfram, "Universality and complexity in cellular automata", Physica 10D (1984) 1.
8. E. R. Berlekamp, J. H. Conway and R. K. Guy, "Winning ways for your mathematical plays", Academic Press (1982), vol. 2, chap. 25; M. Gardner, "Wheels, Life and other mathematical amusements", Freeman (1983).
9. T. Toffoli, "CAM: A high-performance cellular-automaton machine", Physica 10D (1984) 195.
10. J. Guckenheimer and P. Holmes, "Nonlinear oscillations, dynamical systems, and bifurcations of vector fields", Springer (1983).
11. J. E. Hopcroft and J. D. Ullman, "Introduction to automata theory, languages, and computation", Addison-Wesley (1979).
12. S. Wolfram, "Computation theory of cellular automata", to be published in Comm. Math. Phys.
13. L. Hurd, "Formal language theory characterizations of the limiting behaviour of cellular automata", to be published.
14. S. Willson, "On convergence of configurations", Discrete Math. 23 (1978) 279.
15. T. Yaku, "The constructability of a configuration in a cellular automaton", J. Comput. System Sci. 7 (1973) 481; U. Golze, "Differences between 1- and 2-dimensional cell spaces", in A. Lindenmayer and G. Rozenberg (eds.), "Automata, languages, development", North-Holland (1976).
16. J. Hardy, O. de Pazzis and Y. Pomeau, "Molecular dynamics of a classical lattice gas: transport properties and time correlation functions", Phys. Rev. A13 (1976) 1949.
17. S. Willson, "Growth rates and fractional dimensions in cellular automata", Physica 10D (1984) 69.
18. K. Preston et al., "Basics of cellular logic with some applications in medical image processing", Proc. IEEE 67 (1979) 826.
19. P. Grassberger, "Chaos and diffusion in deterministic cellular automata", Physica 10D (1984) 52.
20. S. Wolfram, "Cellular automata", Los Alamos Science (Fall 1983); "Some recent results and questions about cellular automata", Institute for Advanced Study preprint (September 1983).
21. R. Rosensweig, "Fluid dynamics and science of magnetic liquids", Adv. Electronics & Electron Phys. 48 (1979) 103; "Magnetic fluids", Sci. Amer. 247 (1982) no. 4, 136.
22. G. A. Hedlund, "Endomorphisms and automorphisms of the shift dynamical system", Math. Systems Theory 3 (1969) 320; G. A. Hedlund, "Transformations commuting with the shift", in "Topological dynamics", ed. J. Auslander and W. H. Gottschalk, Benjamin (1968); S. Amoroso and Y. N. Patt, "Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures", J. Comp. & Sys. Sci. 6 (1972) 448; M. Nasu, "Local maps

inducing surjective global maps of one-dimensional tessellation automata", Math. Systems Theory 11 (1978) 327.

23. R. Berger, "The undecidability of the domino problem", Mem. Amer. Math. Soc., no. 66 (1966); R. Robinson, "Undecidability and nonperiodicity for tilings of the plane", Inventiones Math. 12 (1971) 177.

24. N. Packard, "Complexity of growing patterns in cellular automata", Institute for Advanced Study preprint (October 1983).

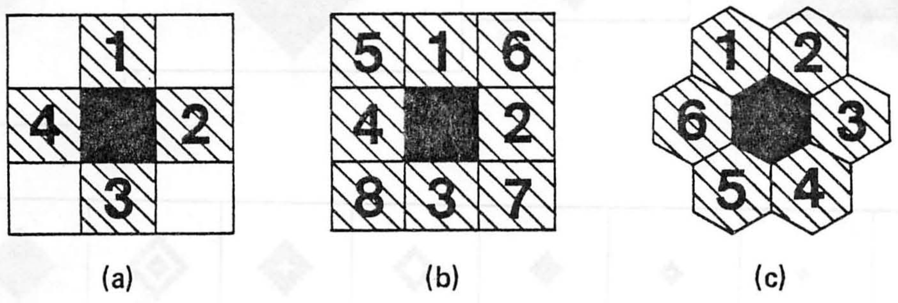
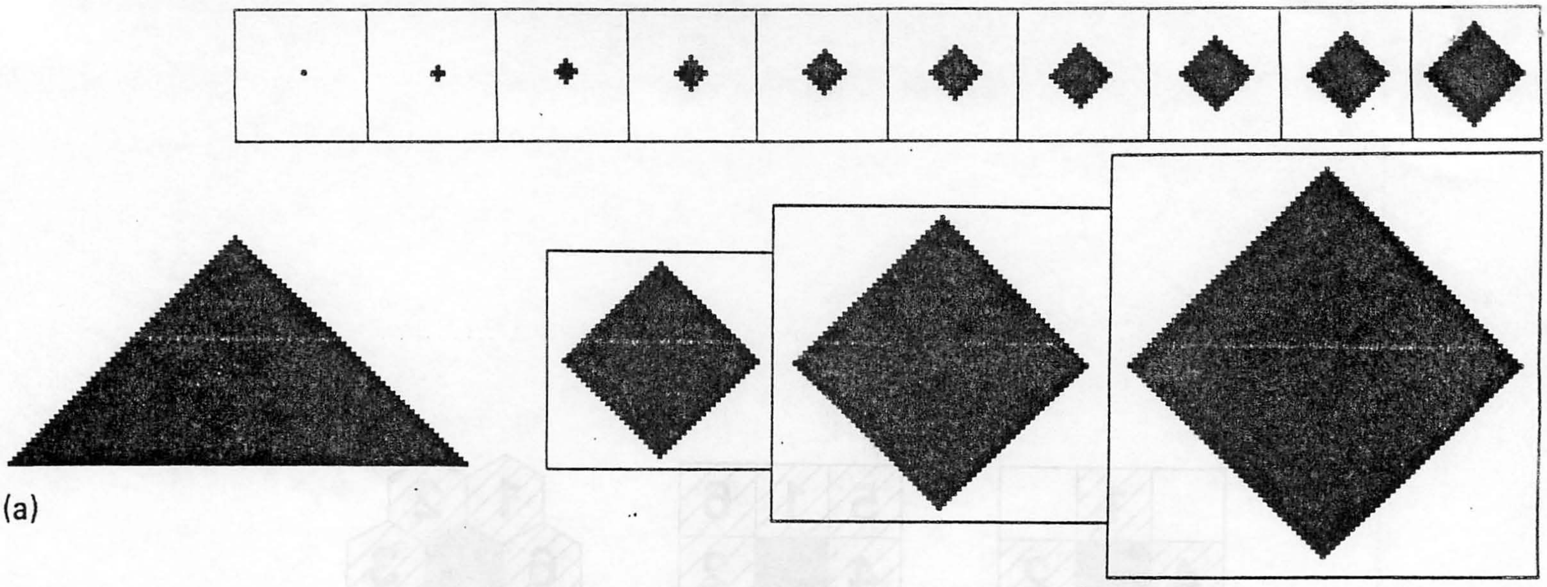
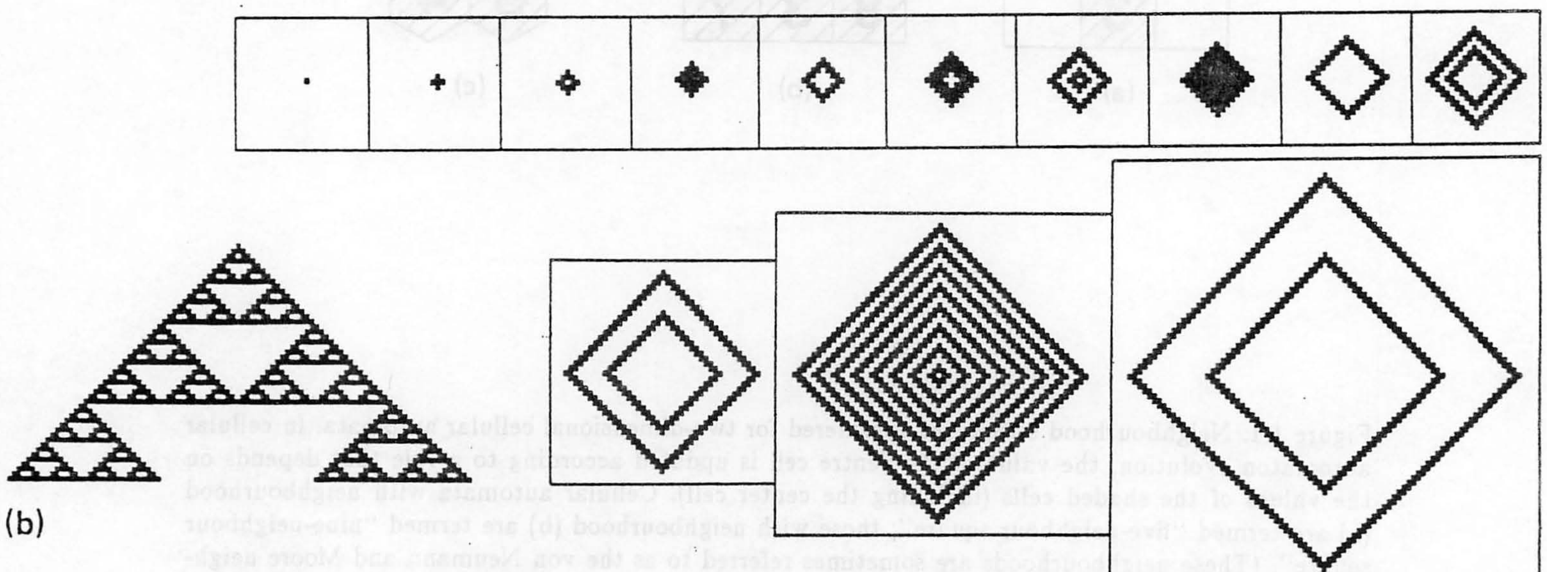
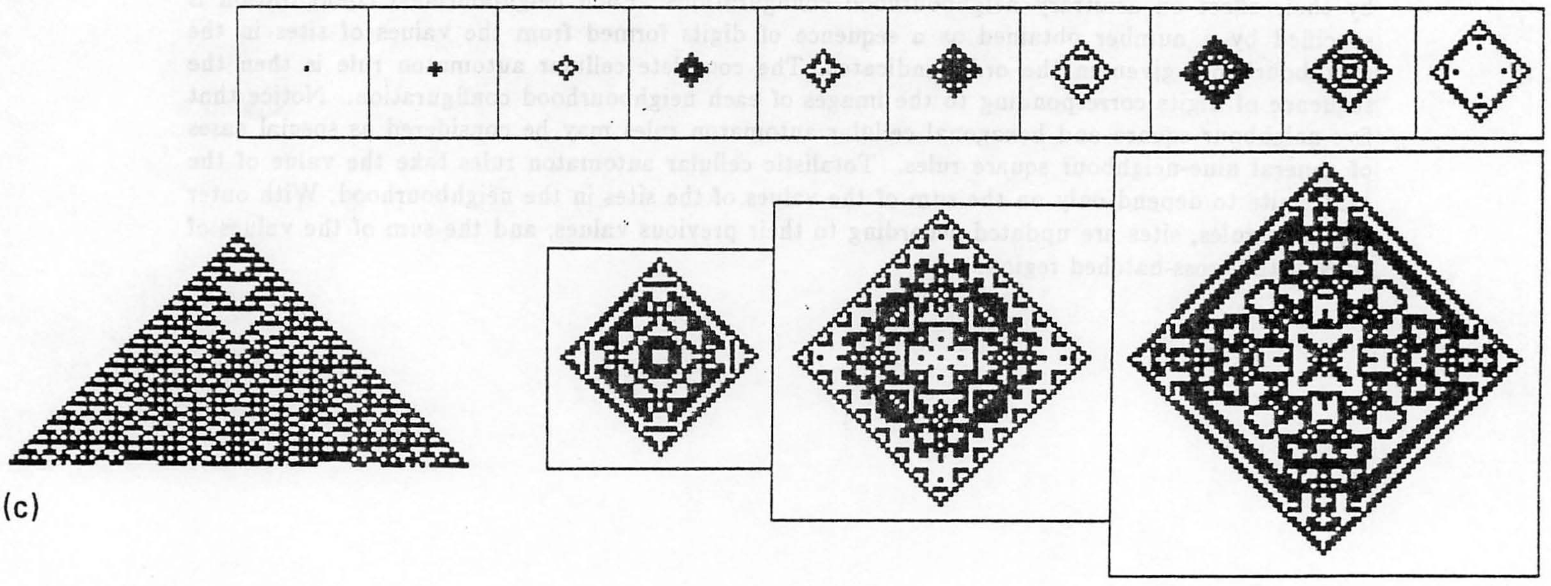
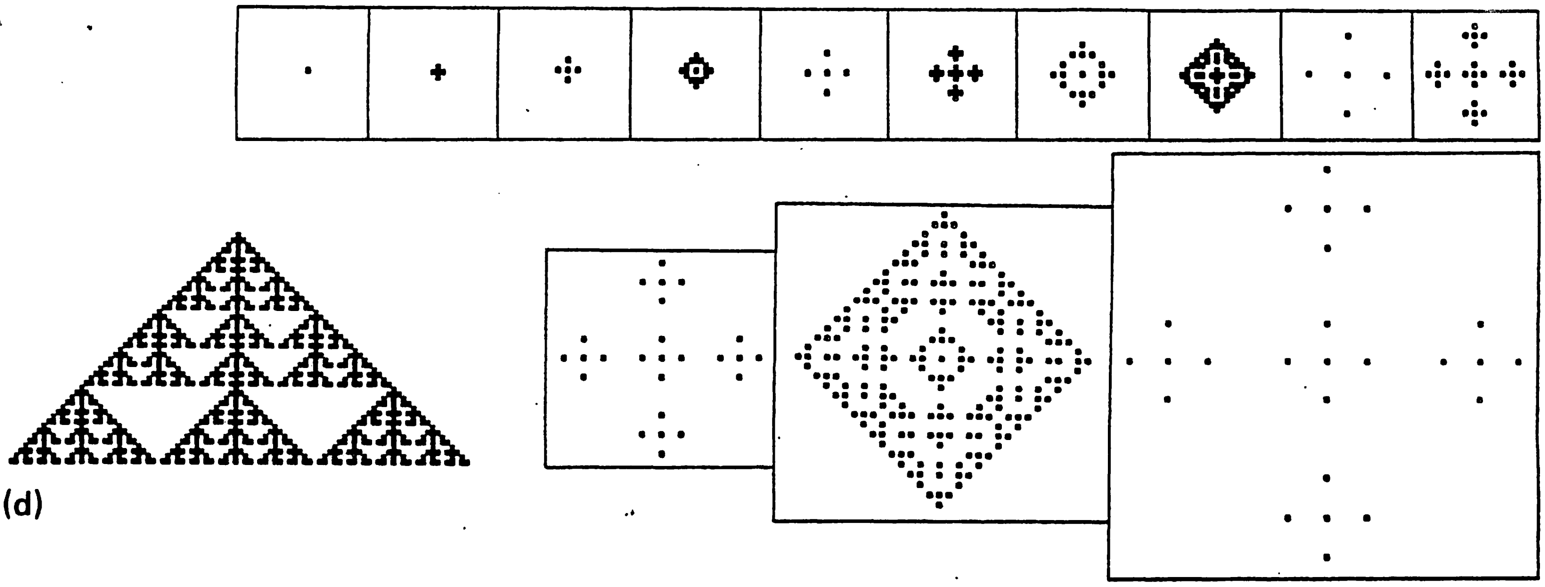


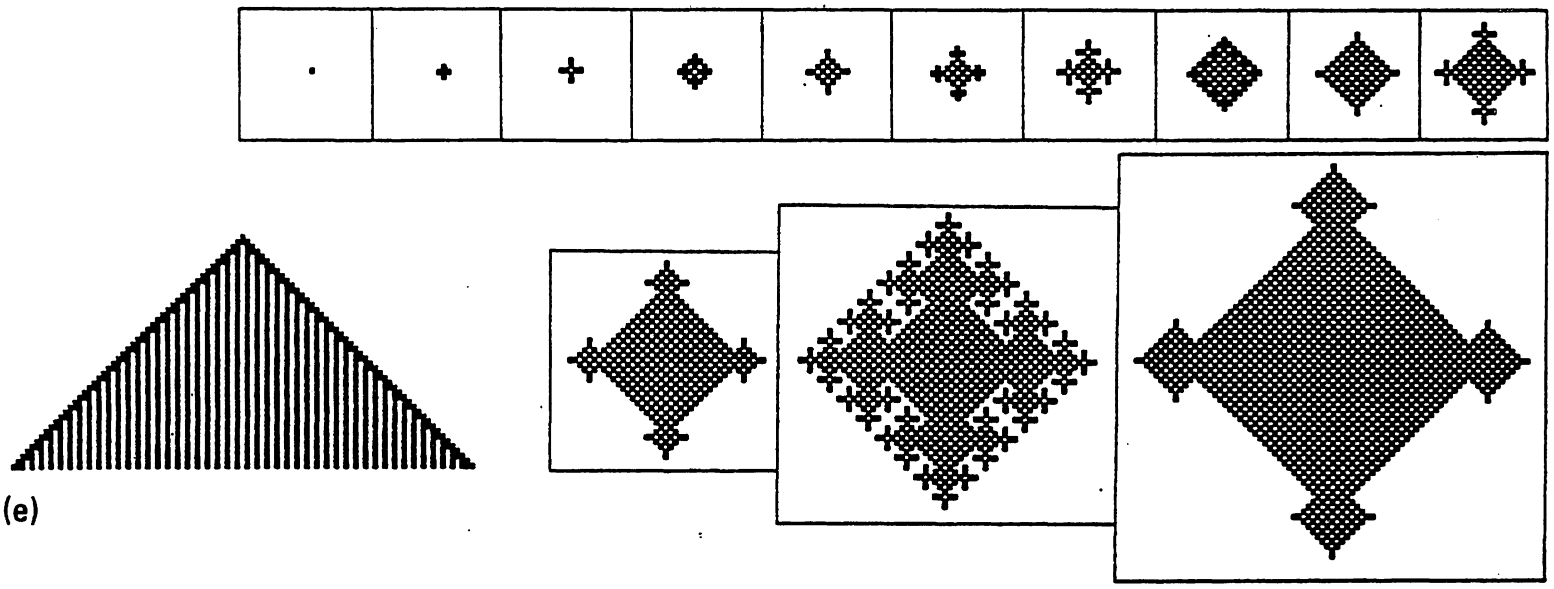
Figure 1.1: Neighbourhood structures considered for two-dimensional cellular automata. In cellular automaton evolution, the value of the centre cell is updated according to a rule that depends on the values of the shaded cells (including the center cell). Cellular automata with neighbourhood (a) are termed "five-neighbour square"; those with neighbourhood (b) are termed "nine-neighbour square". (These neighbourhoods are sometimes referred to as the von Neumann and Moore neighbourhoods, respectively.) Those with neighbourhood (c) are termed "hexagonal". Triangular lattices are also possible, but are not considered here. General cellular automaton rules are specified by their effect on arbitrary neighbourhood configurations. Each neighbourhood configuration is specified by a number obtained as a sequence of digits formed from the values of sites in the neighbourhood, given in the order indicated. The complete cellular automaton rule is then the sequence of digits corresponding to the images of each neighbourhood configuration. Notice that five-neighbour square and hexagonal cellular automaton rules may be considered as special cases of general nine-neighbour square rules. Totalistic cellular automaton rules take the value of the centre site to depend only on the sum of the values of the sites in the neighbourhood. With outer totalistic rules, sites are updated according to their previous values, and the sum of the values of sites in the cross-hatched region.

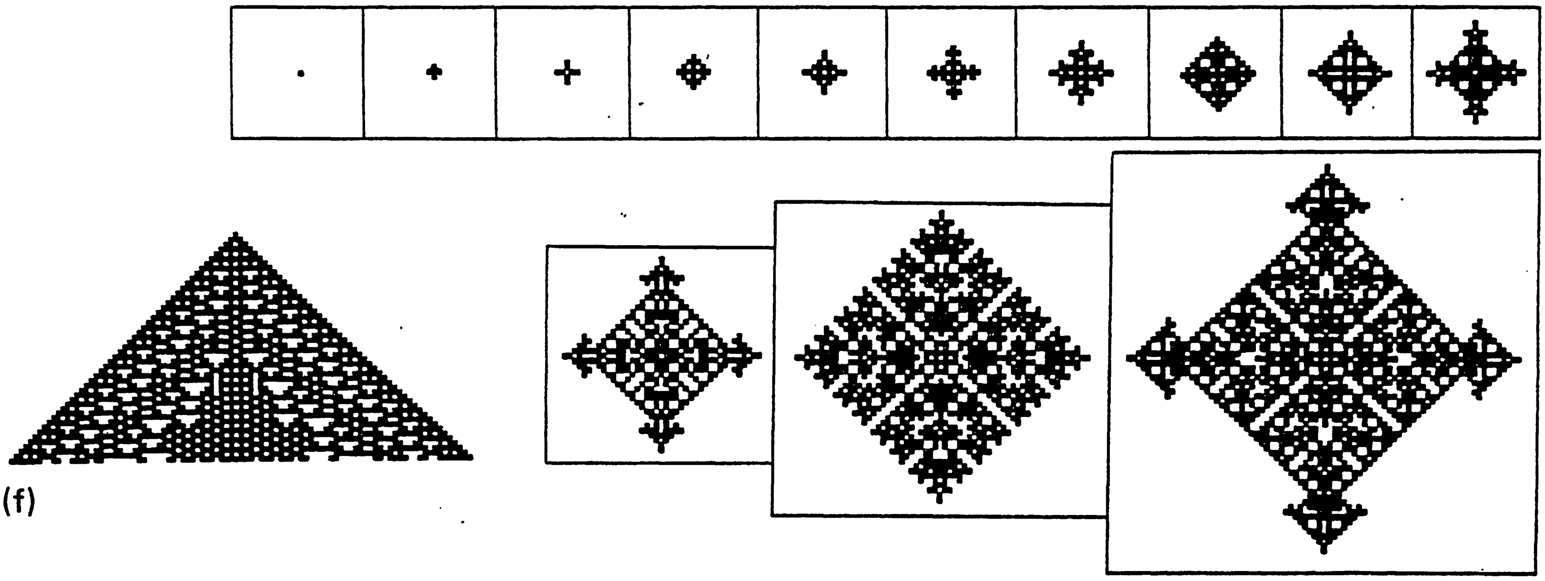

(a)


(b)


(c)


(d)


(e)


(f)

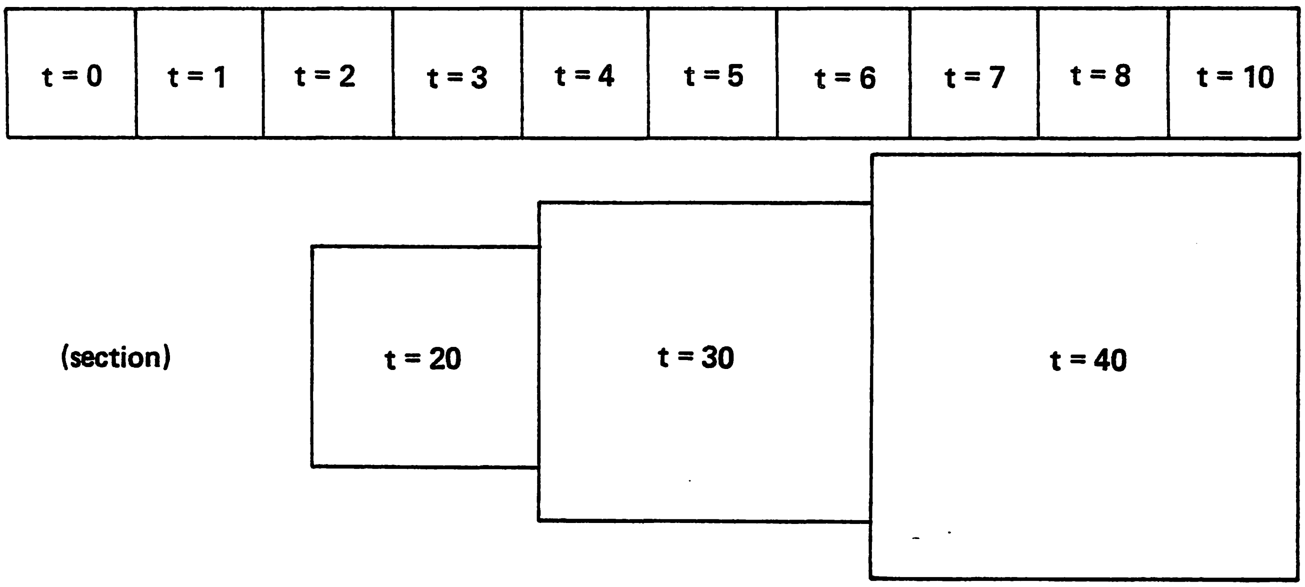


Figure 2.1: Examples of classes of patterns generated by evolution of two-dimensional cellular automata from a single site seed. Each part corresponds to a different cellular automaton rule. All the rules shown are both rotation and reflection symmetric. For each rule, a sequence of frames shows the two-dimensional configurations generated by the cellular automaton evolution after the indicated number of time steps. Black squares represent sites with value 1; white squares sites with value 0. On the left is a spacetime section showing the time evolution of the centre horizontal line of sites in the two-dimensional lattice. Successive lines correspond to successive time steps. The cellular automaton rules shown are five-neighbour square outer totalistic, with codes (a) 1022, (b) 510, (c) 374, (d) 614 (sum modulo 2 rule), (e) 174, (f) 494.

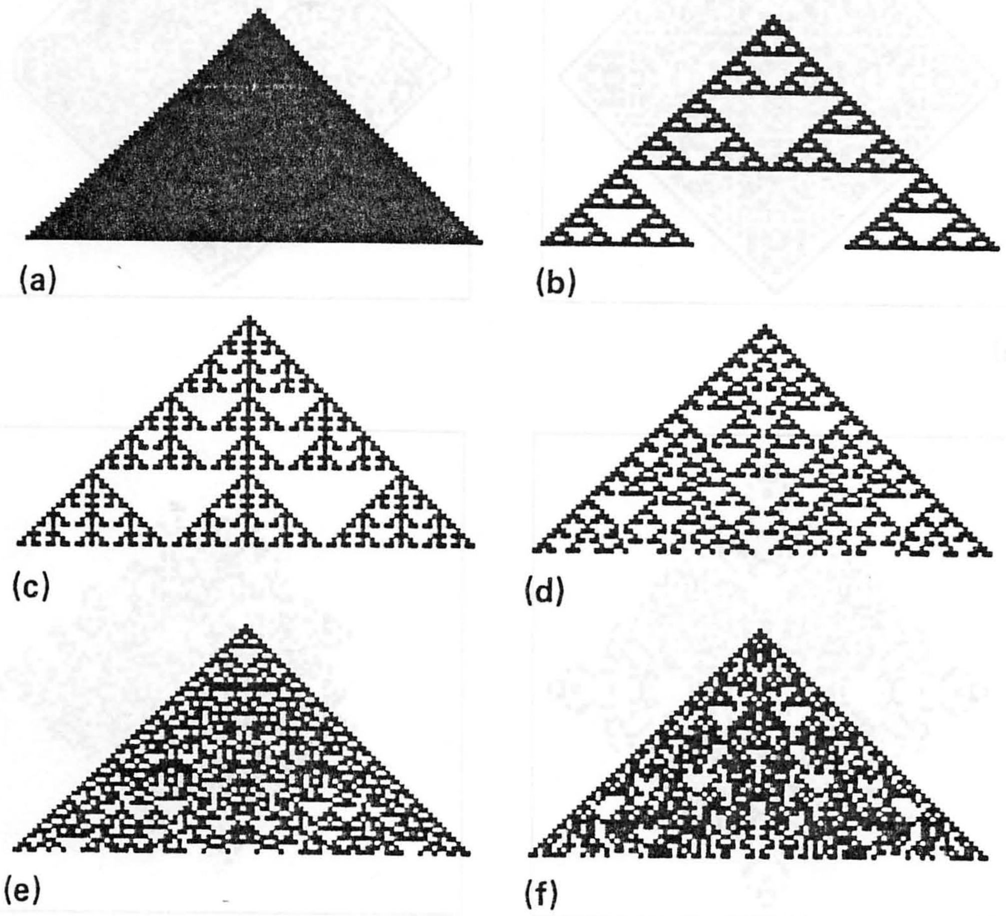
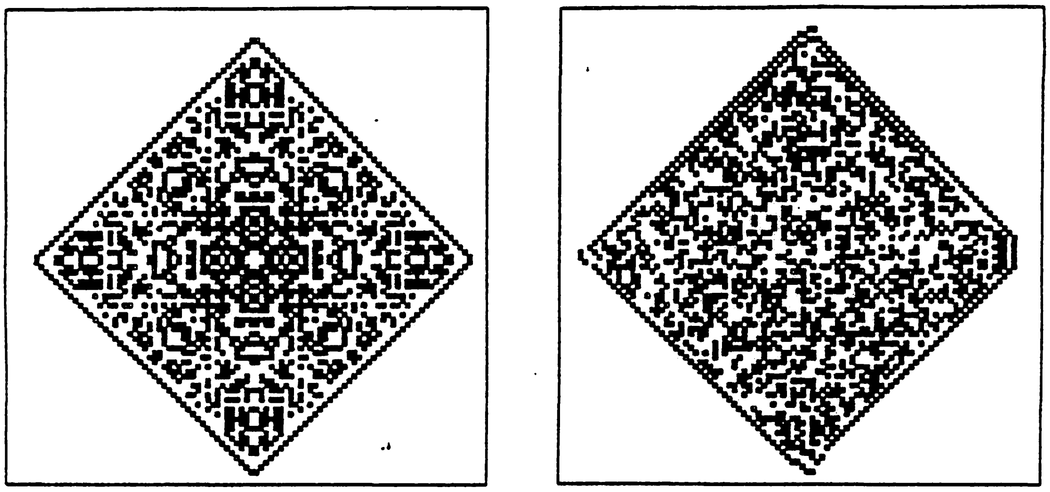
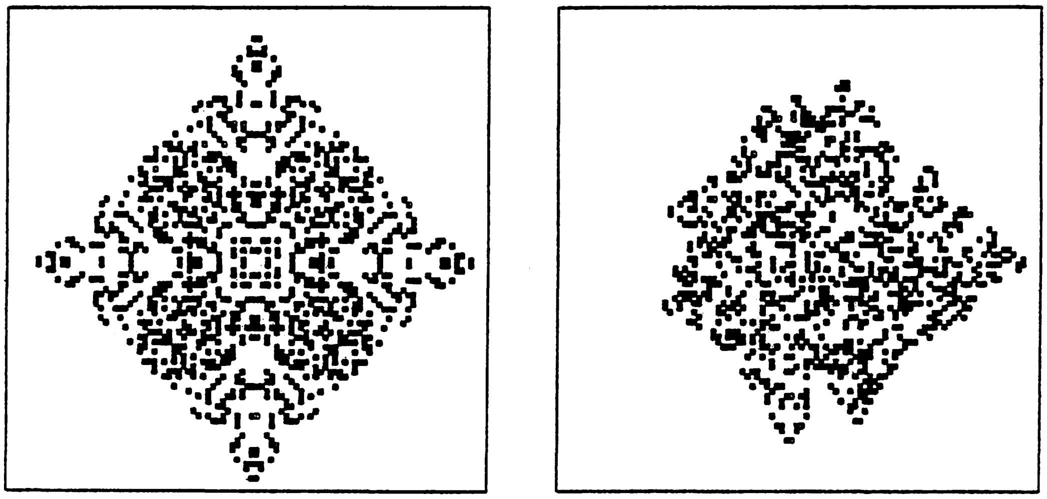
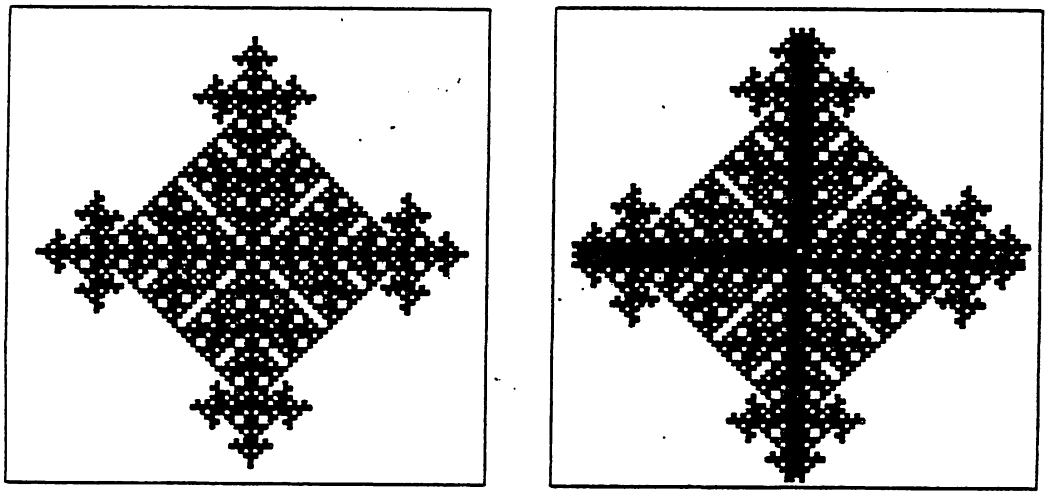
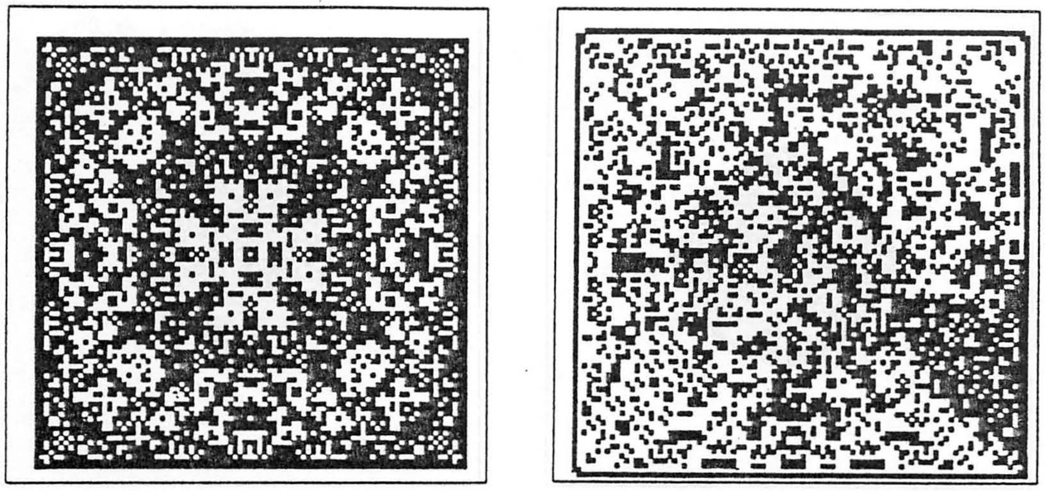


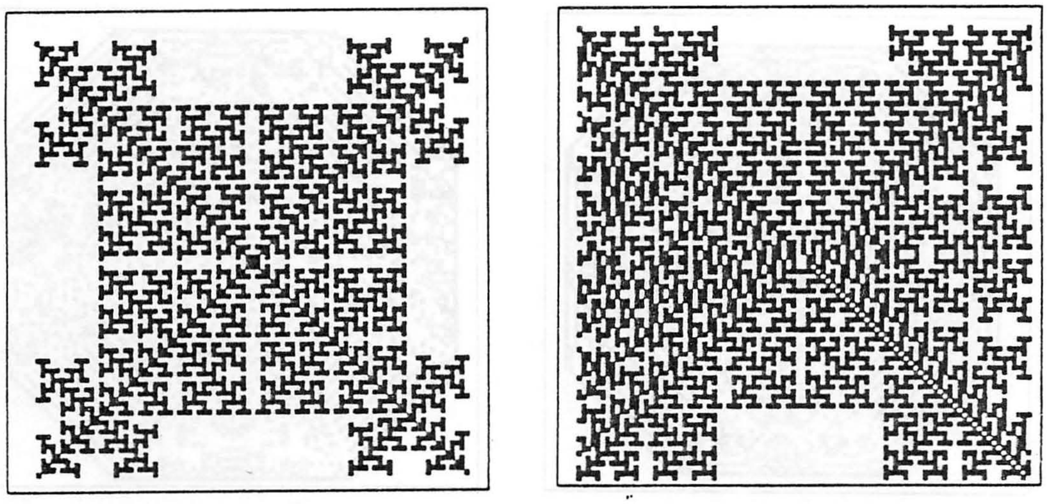
Figure 2.2: Examples of classes of patterns generated by evolution of one-dimensional cellular automata from a single site seed. Successive time steps are shown on successive lines. Nonzero sites are shown black. The cellular automaton rules shown are totalistic nearest-neighbour ($r=1$), with k possible values at each site: (a) $k=2$, code 14, (b) $k=2$, code 6, (c) $k=2$, code 10, (d) $k=3$, code 21, (e) $k=3$, code 102, (f) $k=3$, code 138. Irregular patterns are also generated by some $k=2$, $r=2$ rules (such as that with totalistic code 10), and by asymmetric $k=2$, $r=1$ rules (such as that with rule number 30).

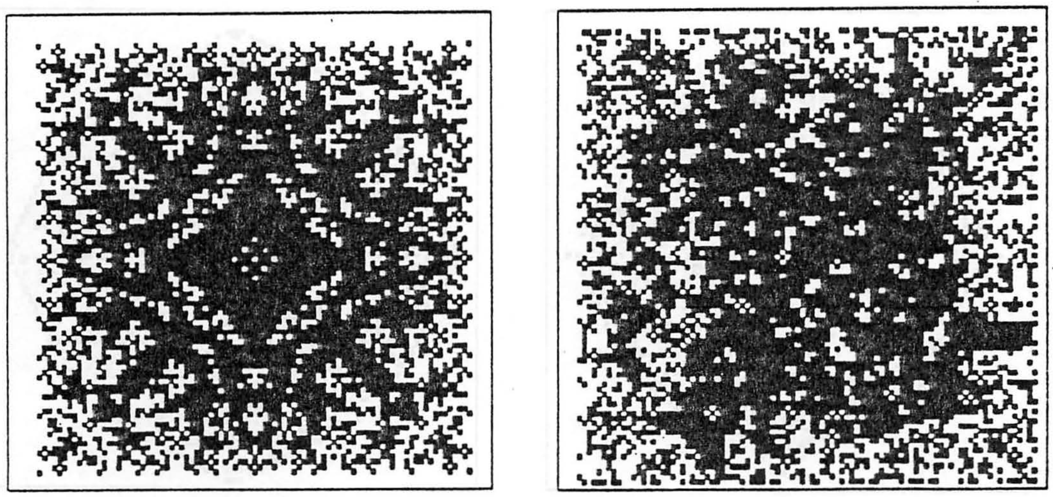

(a)

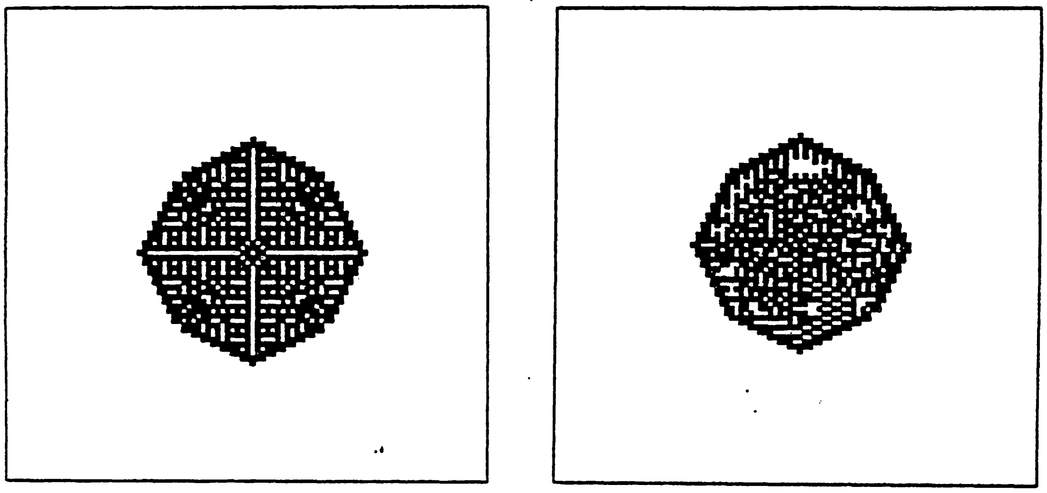

(b)

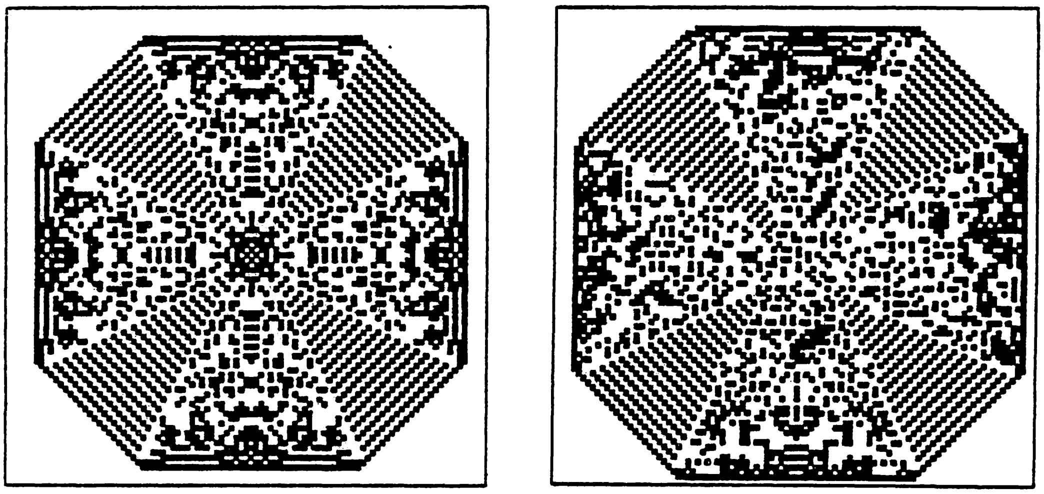

(c)

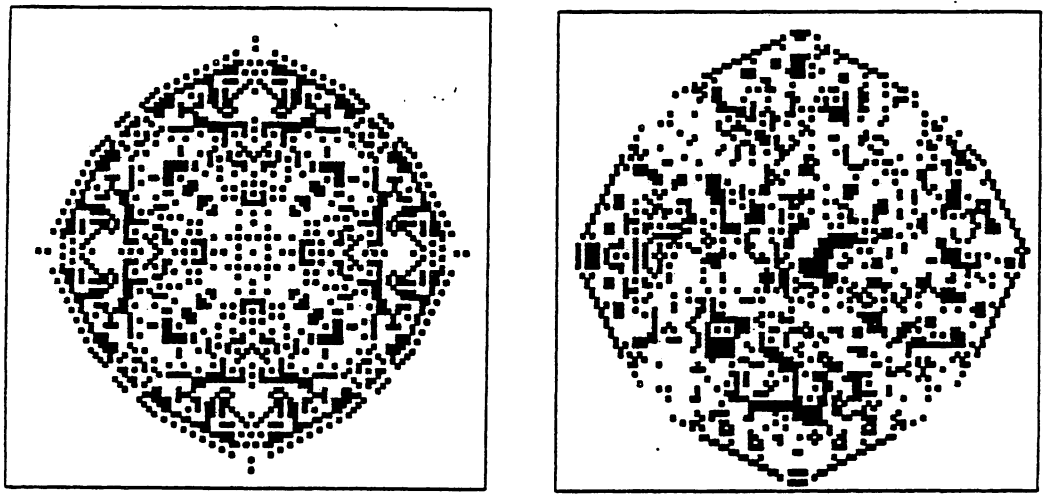

(d)


(e)


(f)


(g)


(h)


(i)

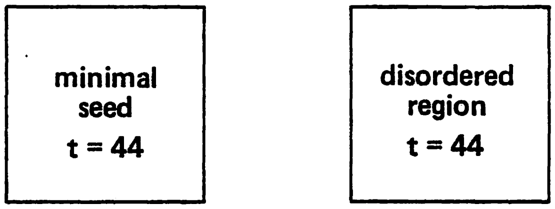


Figure 2.3: Examples of patterns generated by evolution of two-dimensional cellular automata from minimal seeds and small disordered regions. In most cases, growth is initiated by a seed consisting of a single nonzero site; for some of the rules shown, a square of four nonzero sites is required. The cellular automaton rules shown are nine-neighbour square outer totalistic, with codes (a) 143954, (b) 50224, (c) five-neighbour 750, (d) 15822, (e) 699054, (f) 191044, (g) 11202, (h) 93737, (i) 85507.

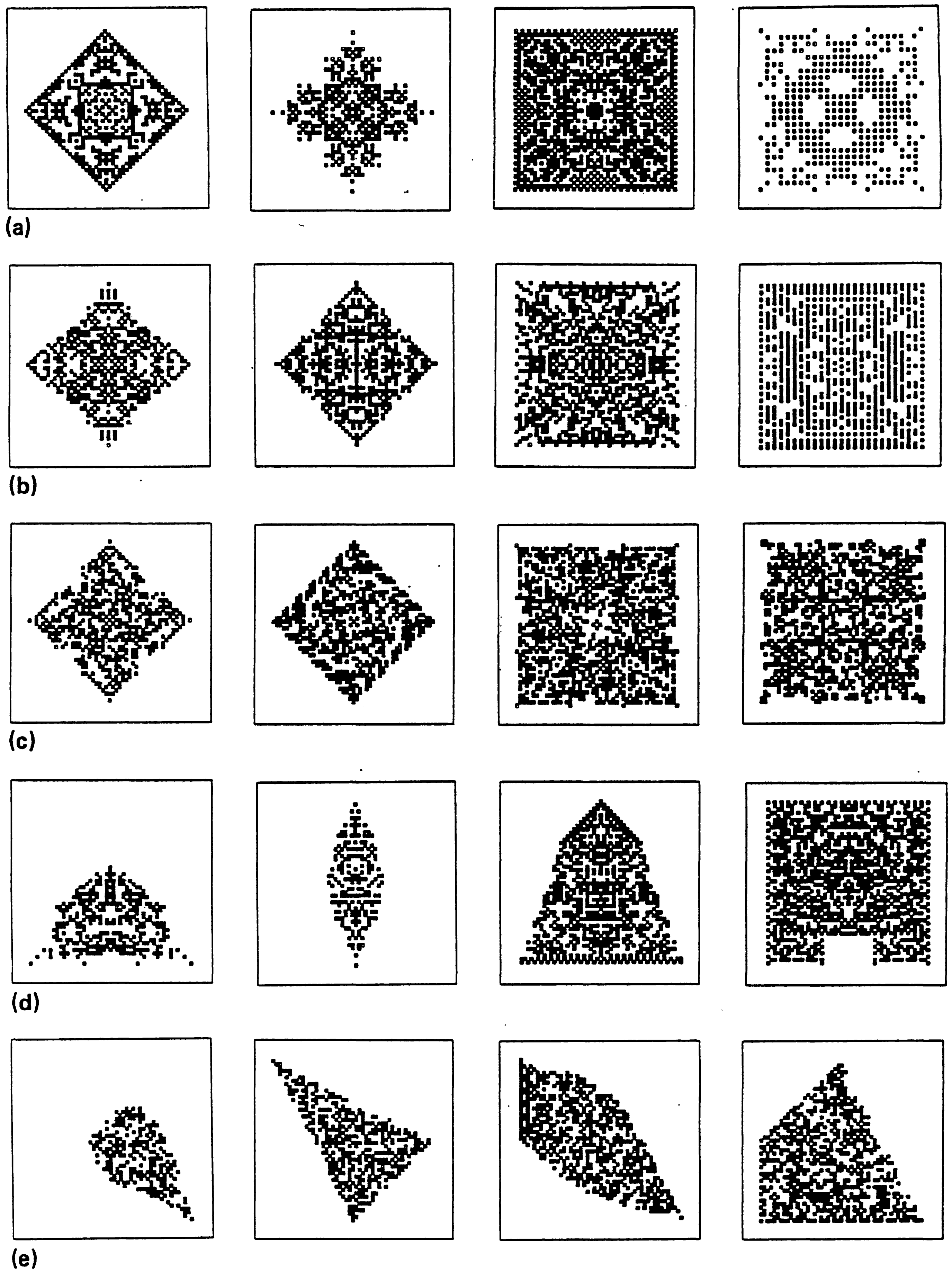
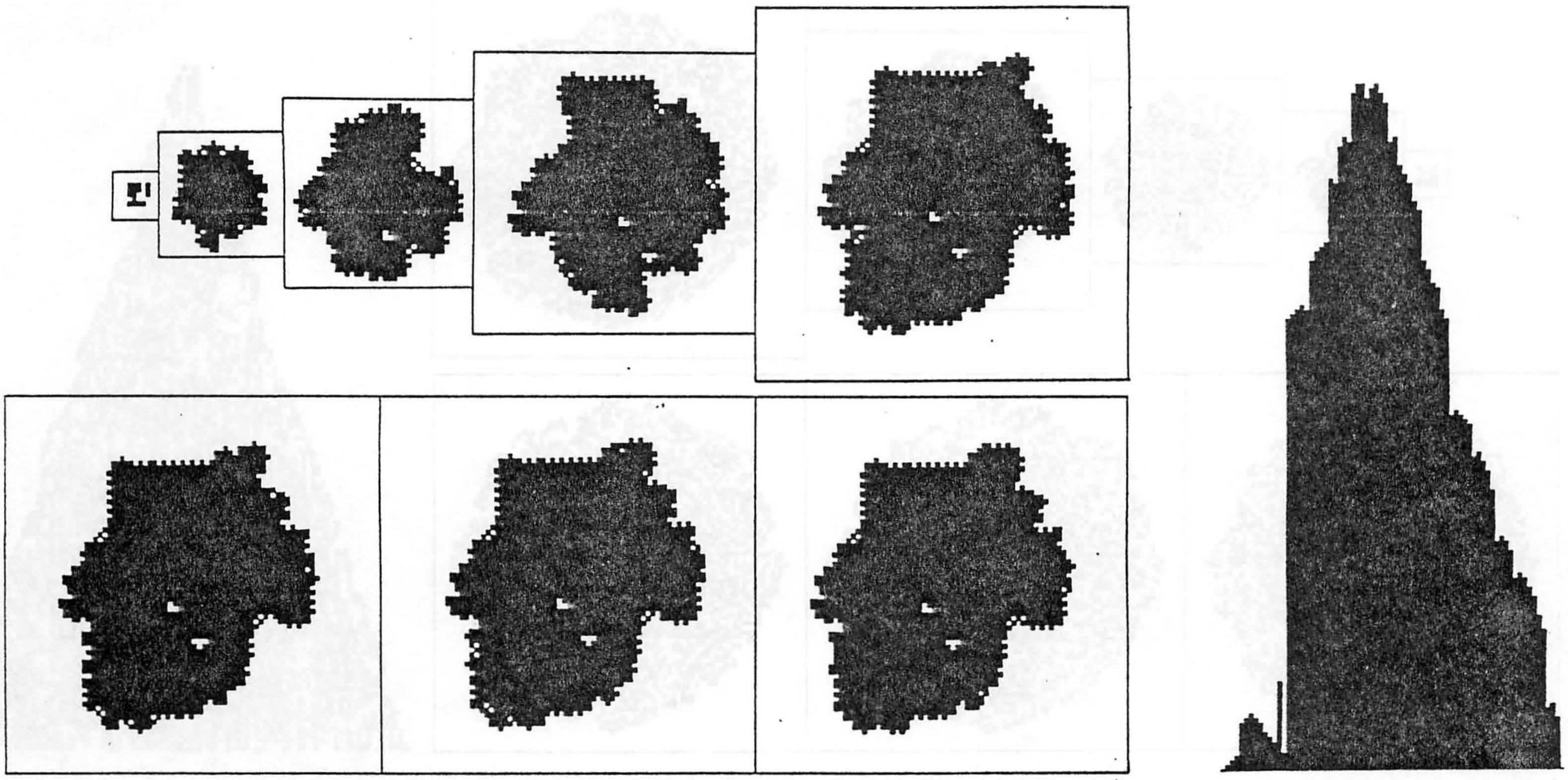
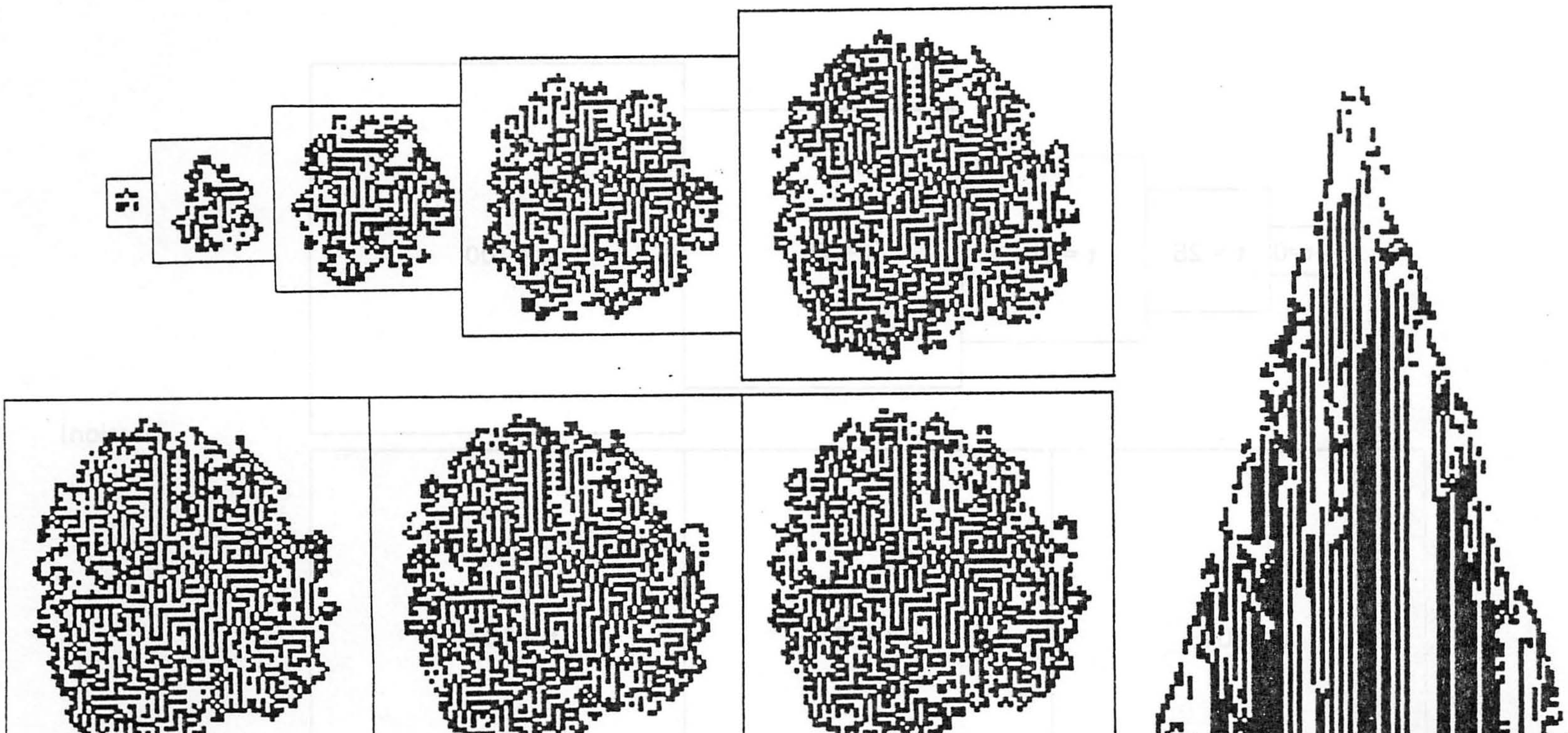
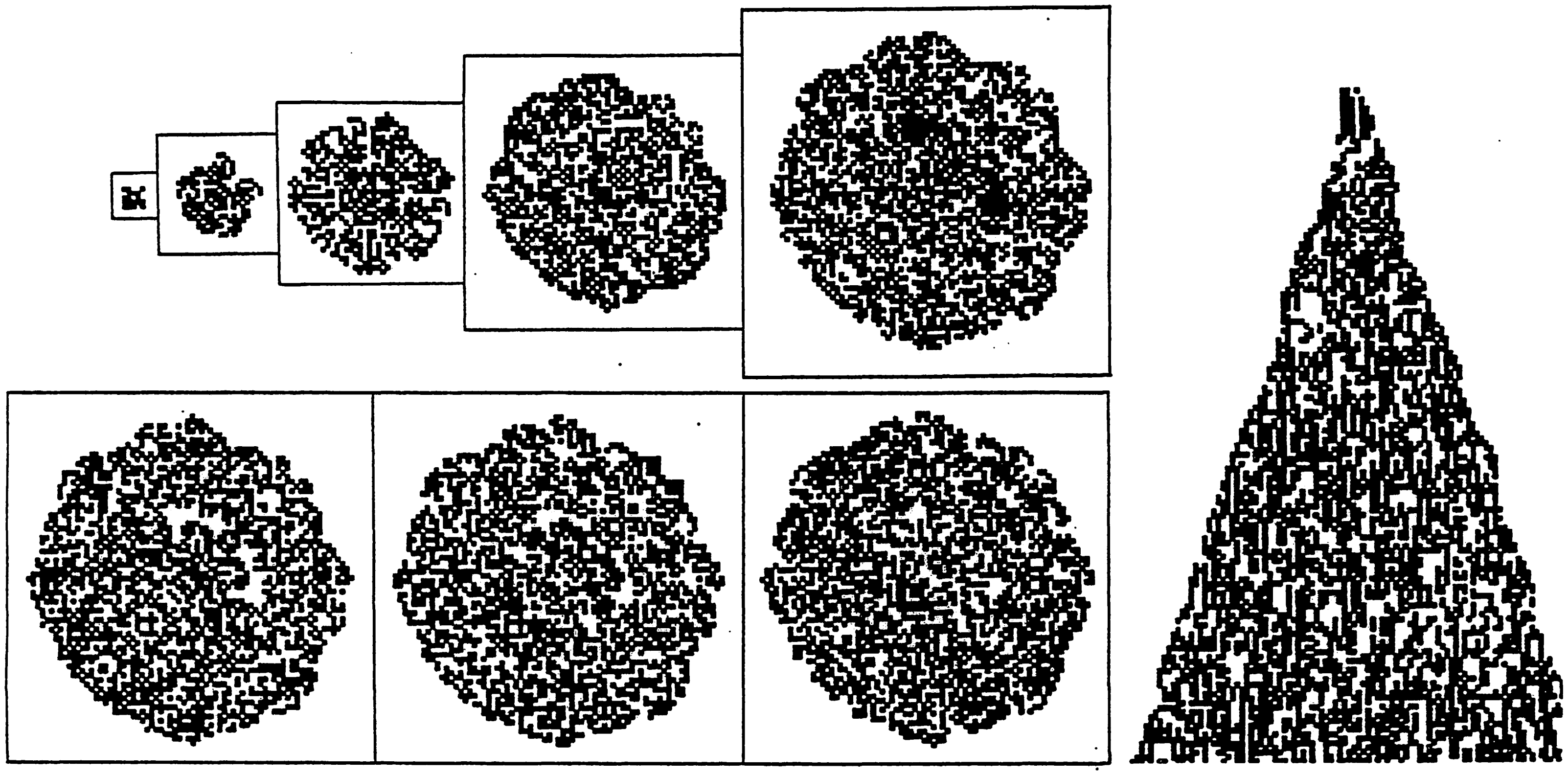


Figure 2.4: Examples of patterns generated by growth from single site seeds for 24 time steps according to general nine-neighbour square rules, with symmetries: (a) all, (b) horizontal and vertical reflection, (c) rotation, (d) vertical reflection, (e) none.


(a)


(b)


(c)

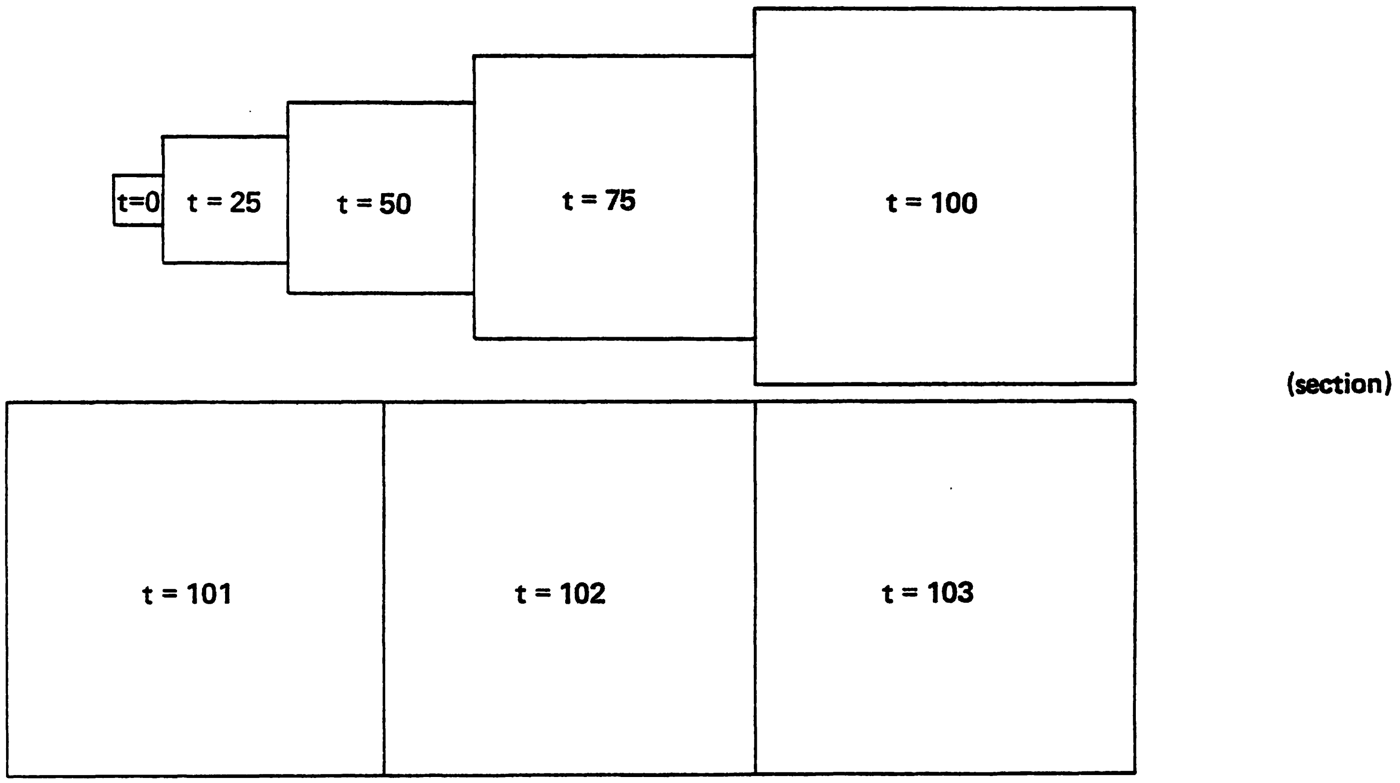


Figure 2.5: Examples of two-dimensional cellular automata that exhibit slow diffusive growth from small disordered regions. The cellular automaton rules shown are nine-neighbour square outer totalistic, with codes (a) 256746, (b) 736, (c) 291552.

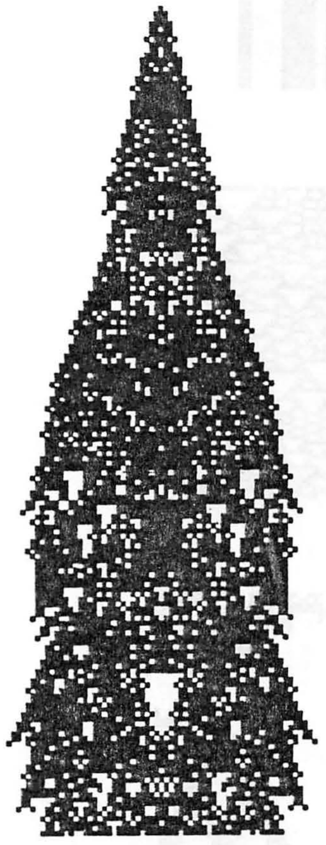
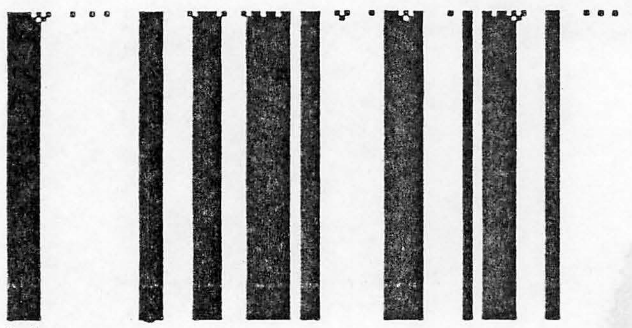
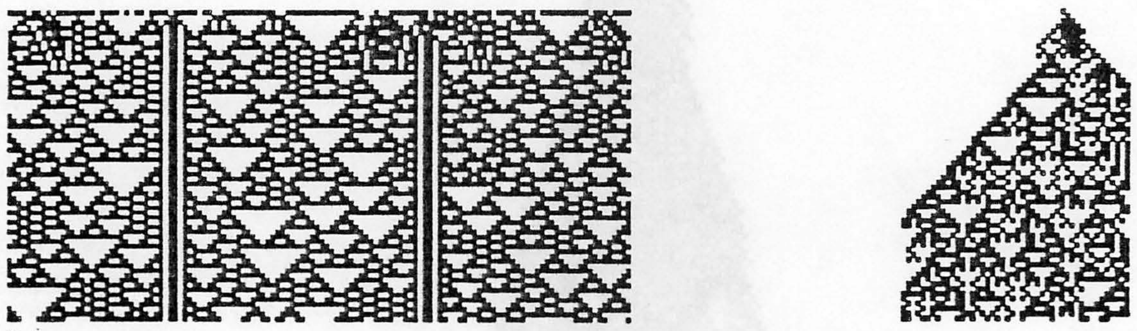
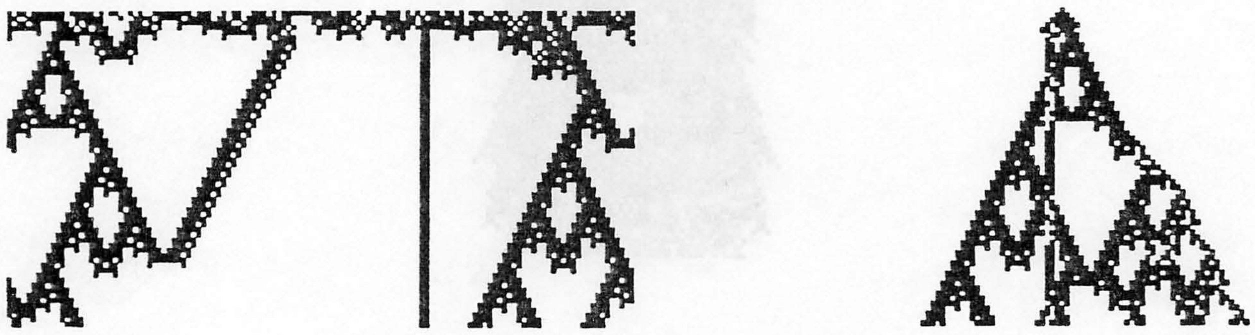
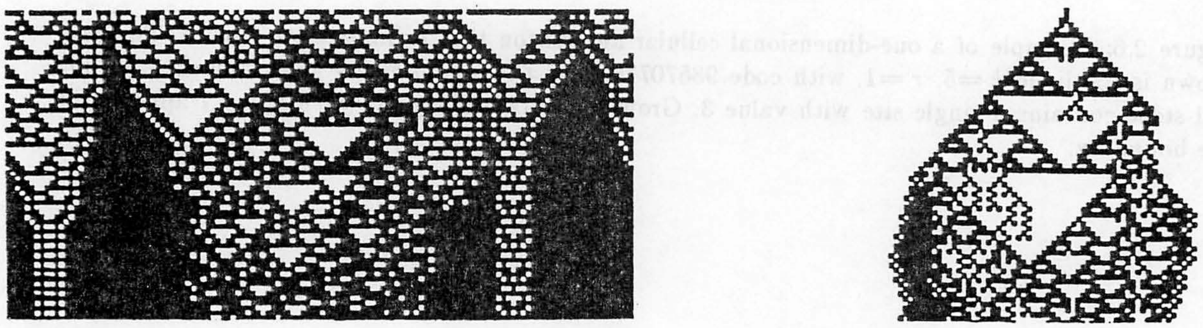


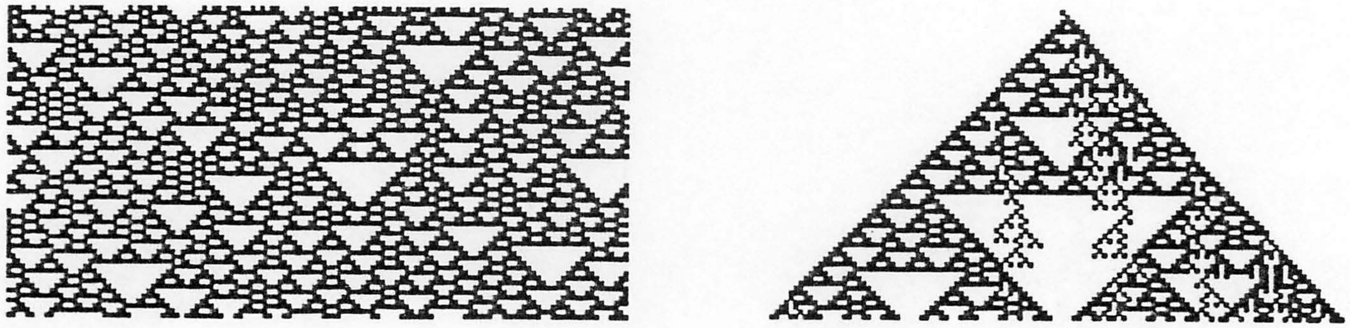
Figure 2.6: Example of a one-dimensional cellular automaton that exhibits slow growth. The rule shown is totalistic $k=5$, $r=1$, with code 985707700. All nonzero sites are shown black. The initial state contains a single site with value 3. Growth occurs when a site with value 1 appears on the boundary.

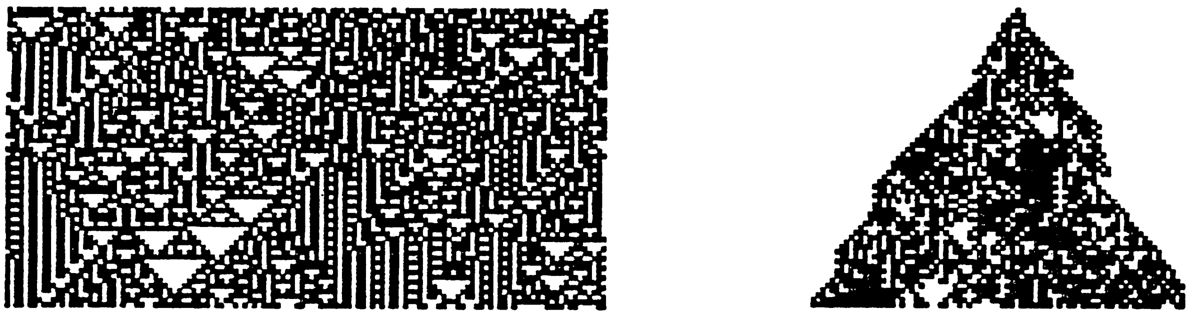

(a)

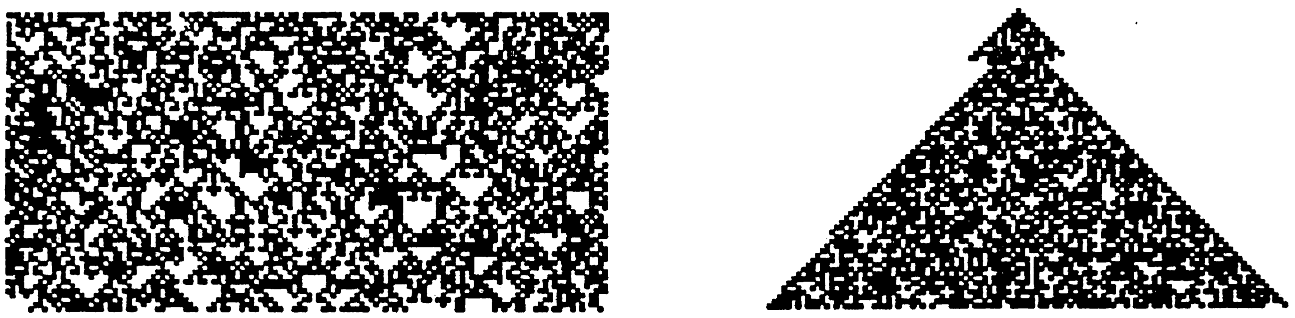

(b)

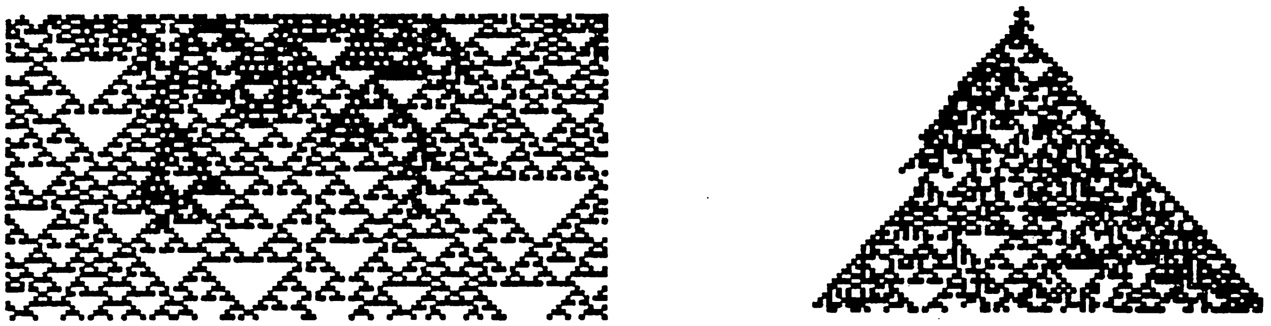

(c)


(d)


(e)


(f)


(g)


(h)

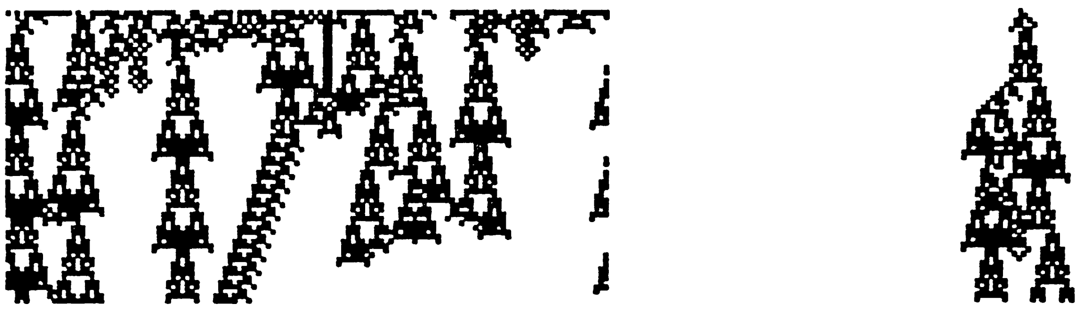
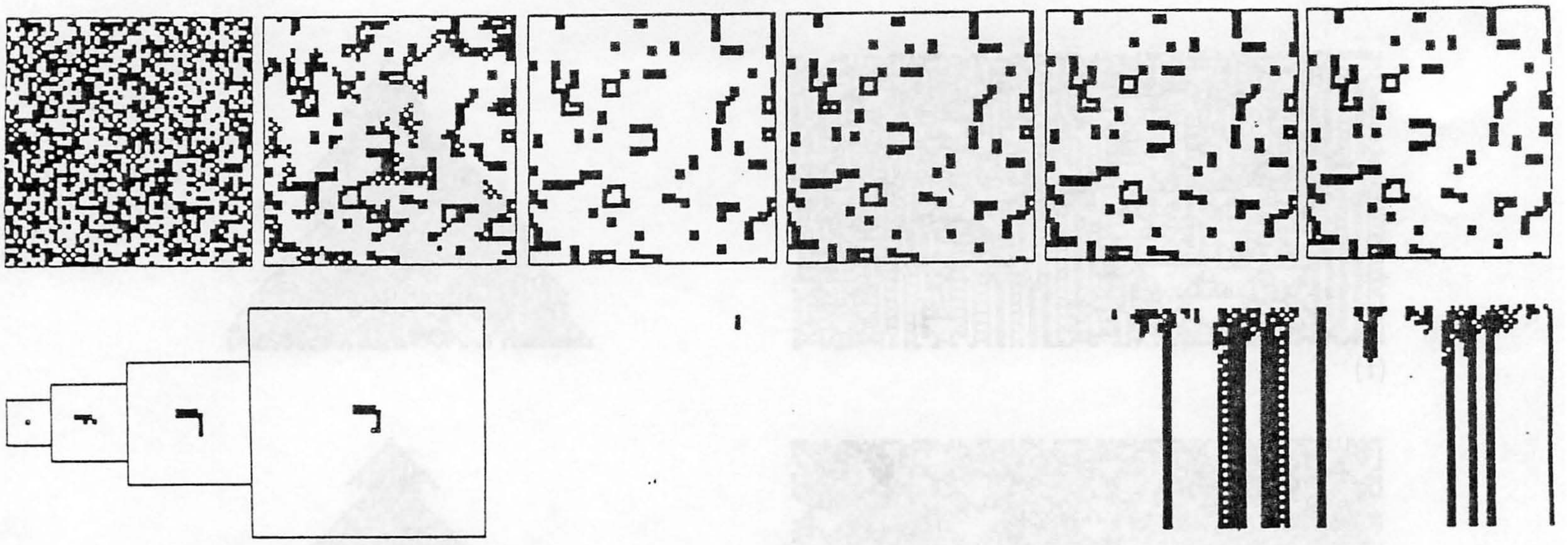
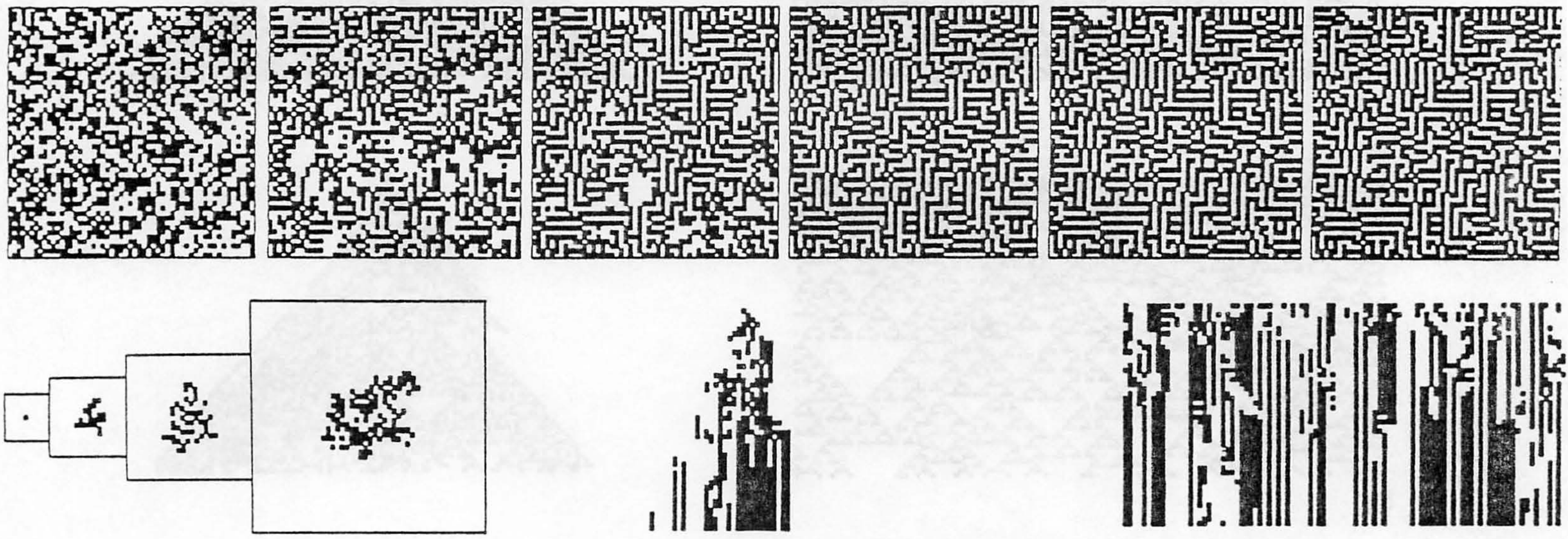
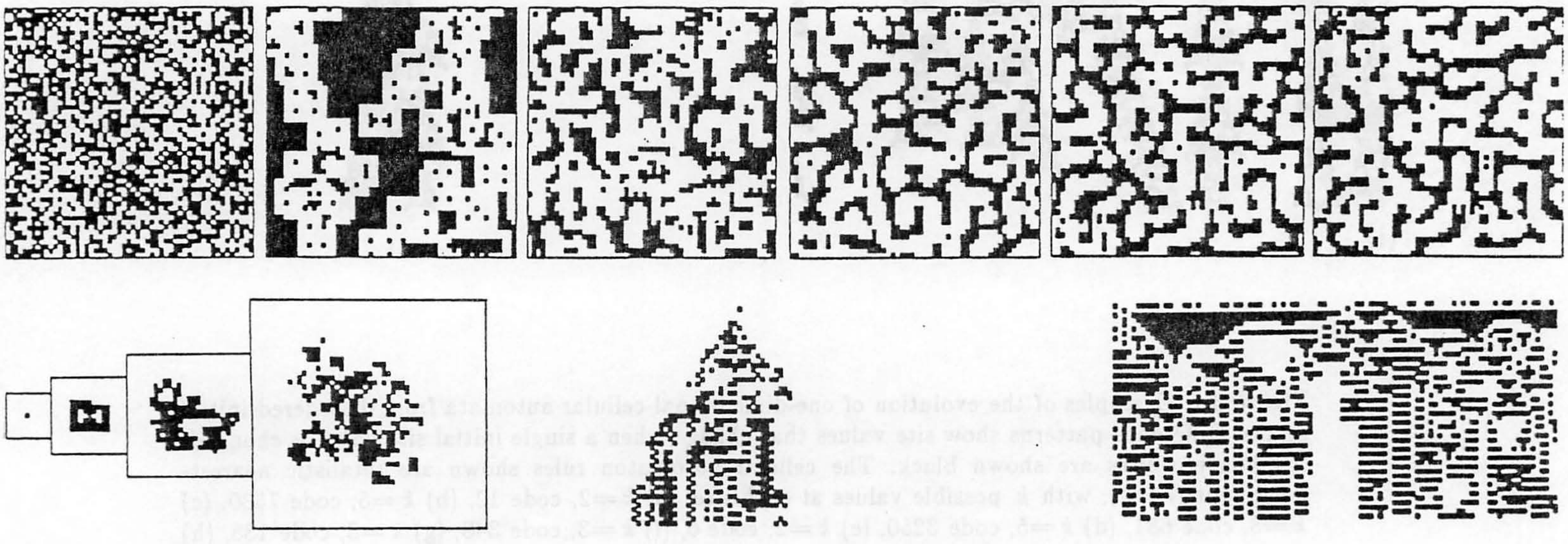
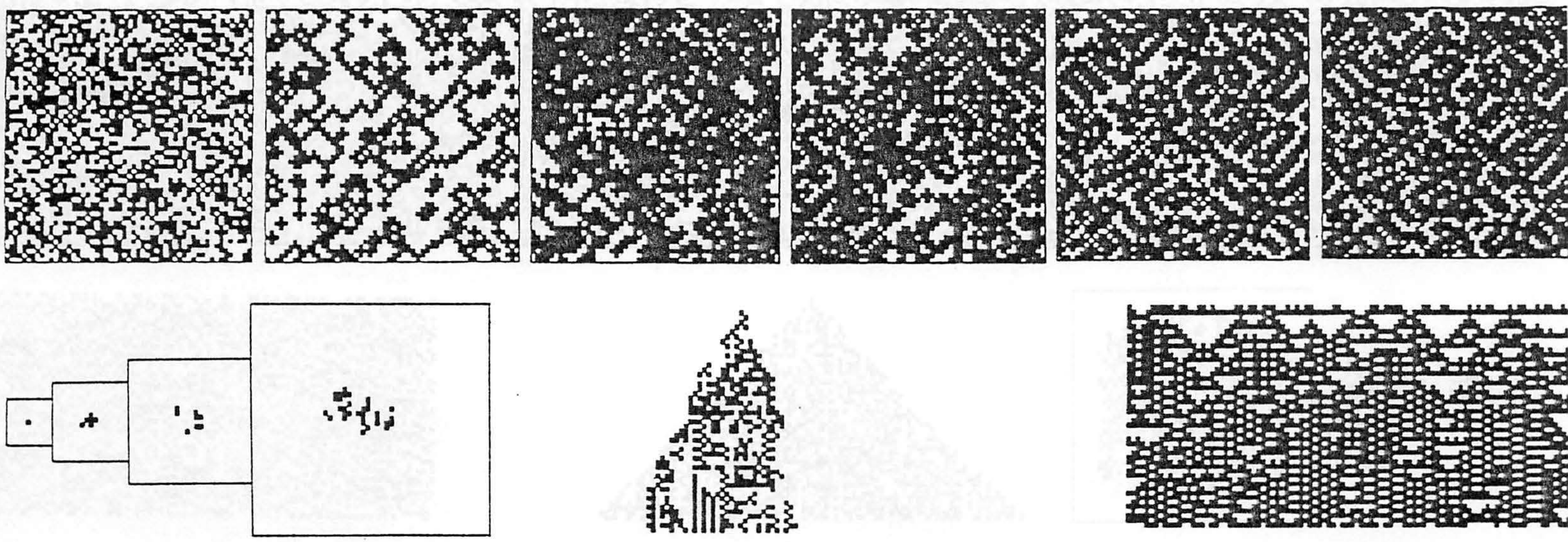

(i)

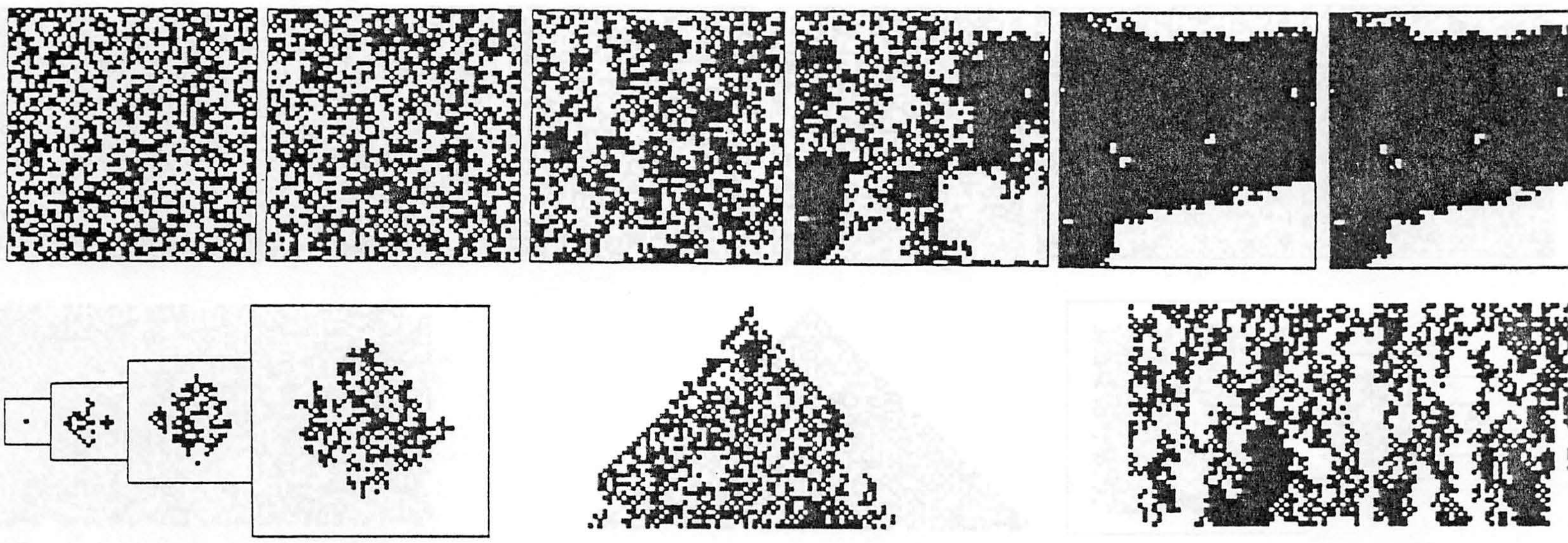
Figure 3.1: Examples of the evolution of one-dimensional cellular automata from disordered initial states. Difference patterns show site values that change when a single initial site value is changed. All nonzero sites are shown black. The cellular automaton rules shown are totalistic nearest-neighbour ($r=1$), with k possible values at each site: (a) $k=2$, code 12, (b) $k=5$, code 7530, (c) $k=3$, code 681, (d) $k=5$, code 3250, (e) $k=2$, code 6, (f) $k=3$, code 348, (g) $k=3$, code 138, (h) $k=3$, code 318, (i) $k=3$, code 792.

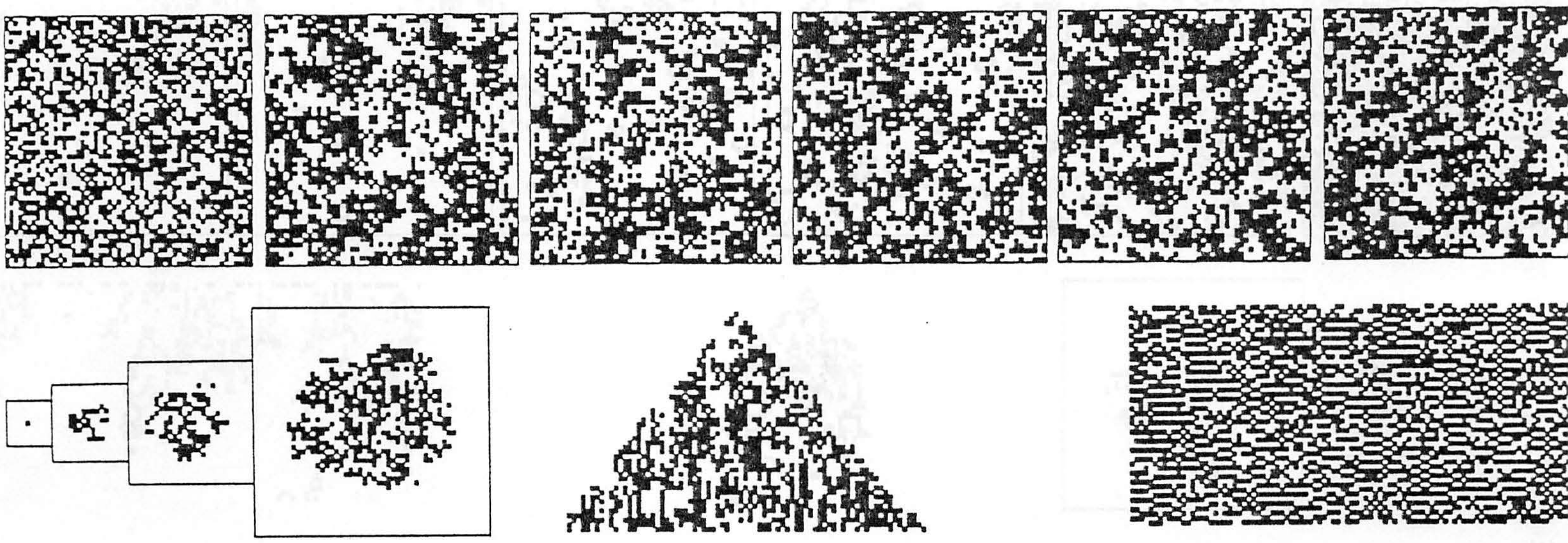

(a)

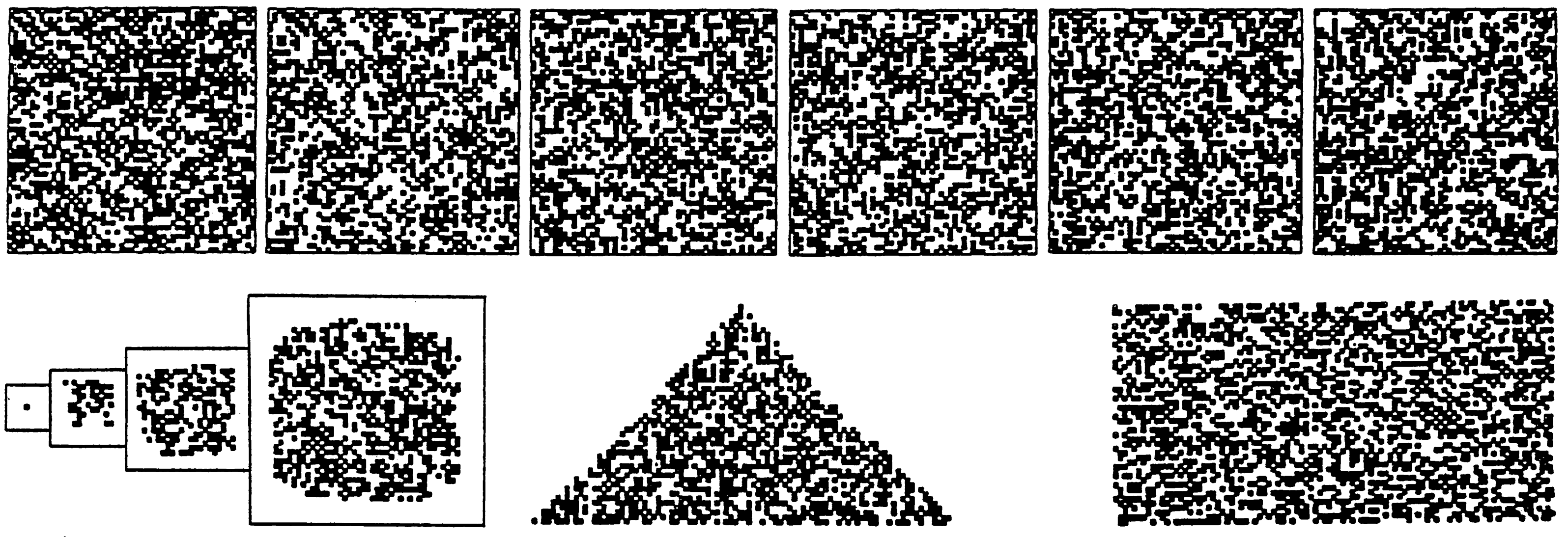

(b)

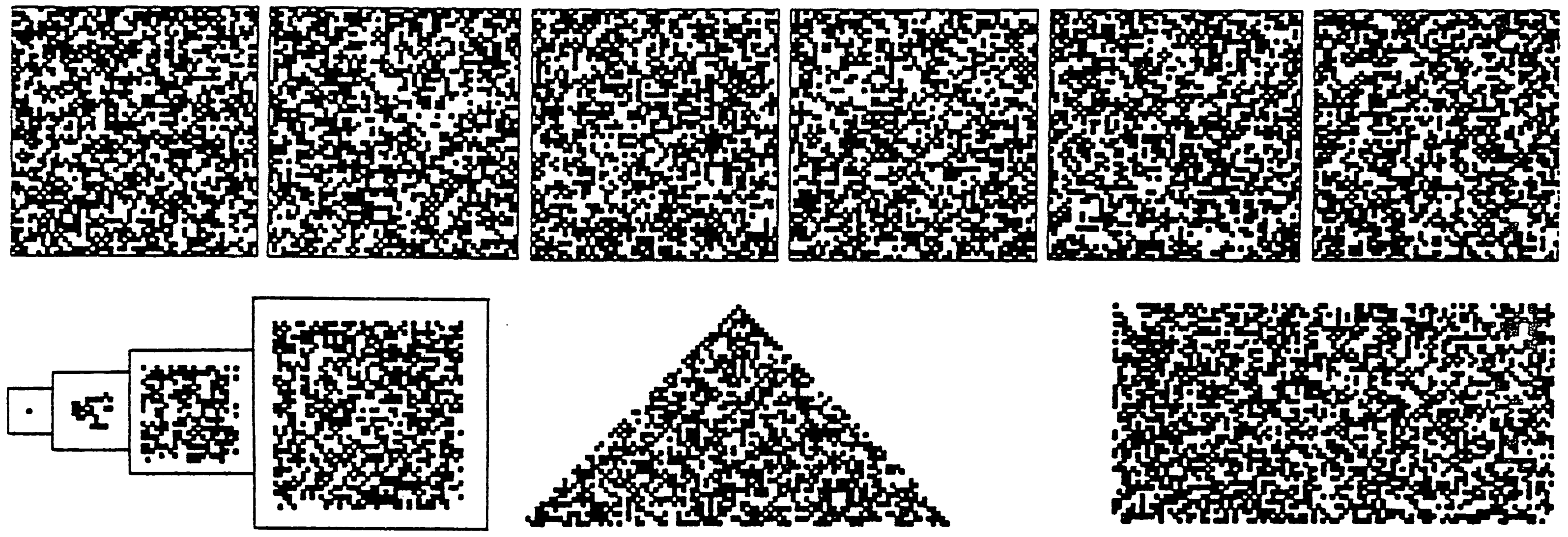

(c)

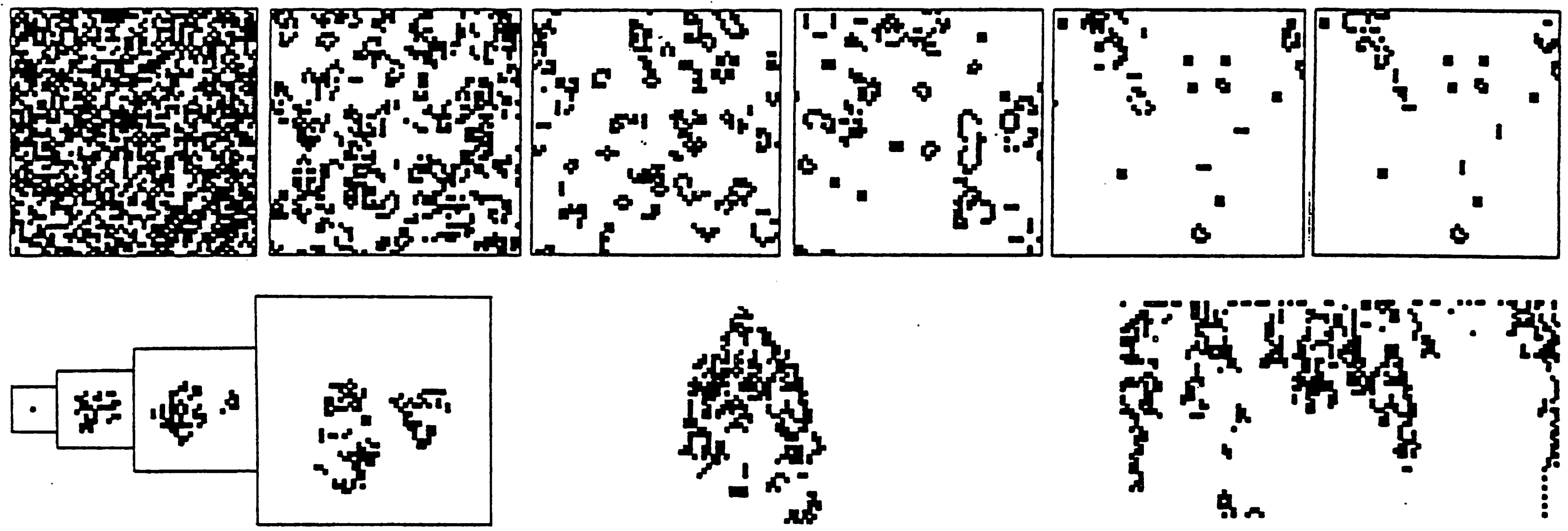

(d)


(e)


(f)


(g)


(h)


(i)

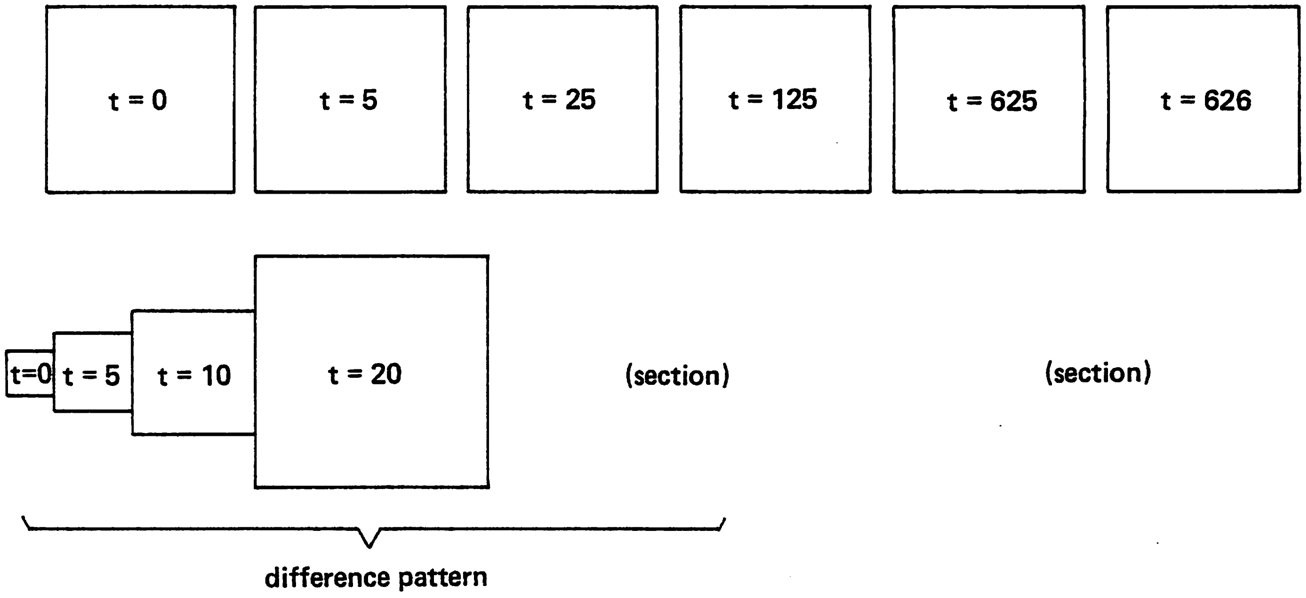
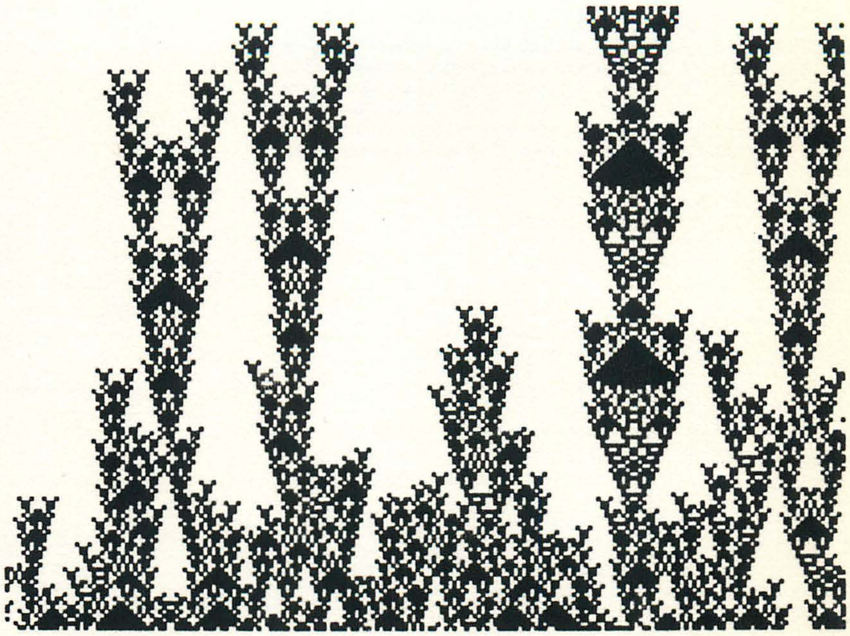
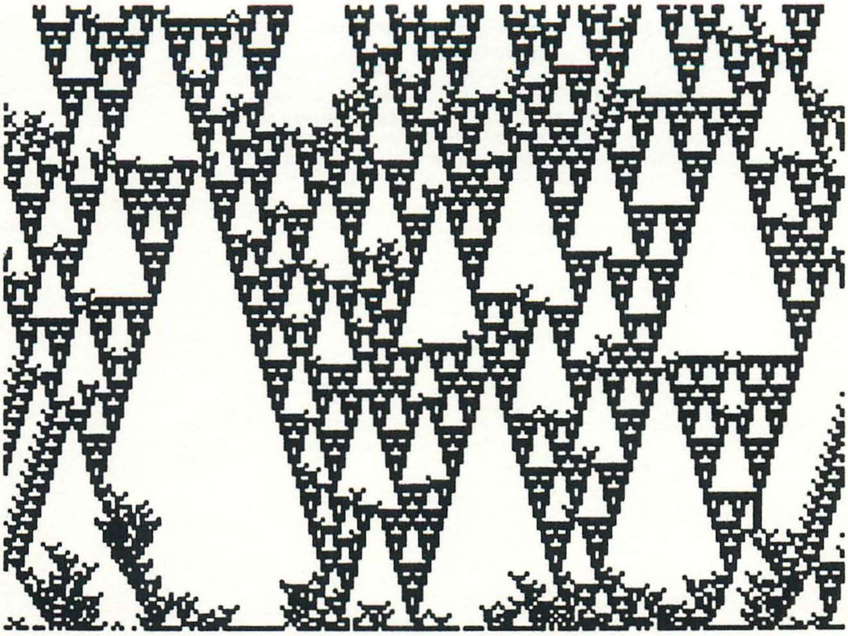


Figure 3.2: Examples of the evolution of two-dimensional cellular automata from disordered initial state. The cellular automaton rules shown are totalistic five-neighbour square with codes: (a) 24, (d) 510, (e) 52; and outer totalistic nine-neighbour with codes: (b) 736, (c) 196623, (f) 152822, (g) 143954, (h) 3276, (i) 224 (the "Game of Life").



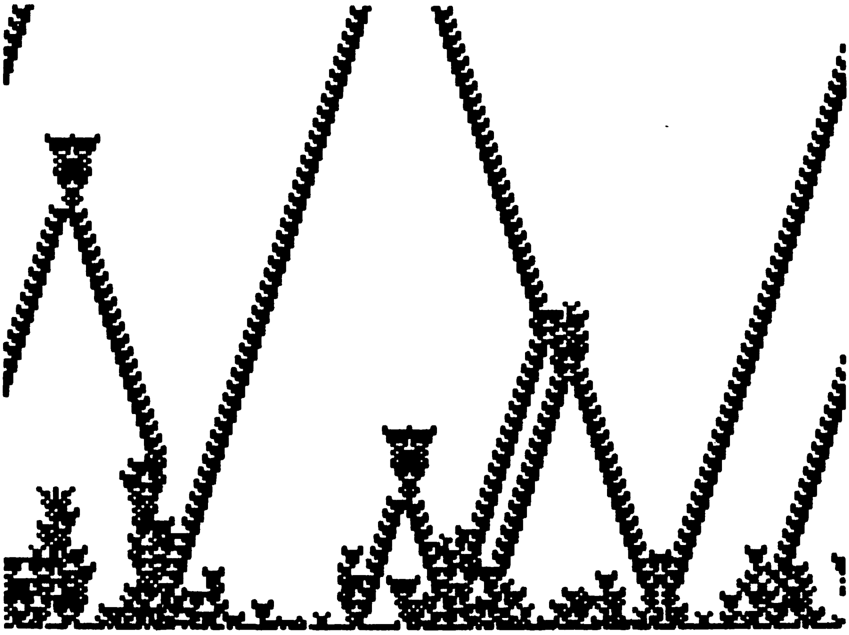
A cellular automaton.

A simple mathematical rule generates each successive line from the last. The points have values zero (white), one (black) or two (black). At each step, each point is updated according to the rule $a_i' = f[a_{i-1} + a_i + a_{i+1}]$, where the nonzero cases are $f[1]=2$, $f[3]=f[4]=f[5]=1$ (code number 357). This cellular automaton is one of a class believed to be capable of universal computation.



A cellular automaton

A simple mathematical rule generates each successive line from the last. The points have values from zero to four; nonzero points are black. At each step, each point is updated according to the rule $a_i' = f[a_{i-1} + a_i + a_{i+1}]$, where the nonzero cases are $f[1]=2$, $f[3]=4$, $f[4]=1$, $f[5]=4$, $f[6]=2$ (code number 44885). This cellular automaton is one of a class believed to be capable of universal computation.



A cellular automaton

A simple mathematical rule generates each successive line from the last. The points have values from zero to four; nonzero points are black. At each step, each point is updated according to the rule $a_i' = f[a_{i-1} + a_i + a_{i+1}]$, where the nonzero cases are $f[1]=4$, $f[3]=1$, $f[4]=2$, $f[5]=1$, $f[6]=4$ (code number 67020). This cellular automaton is one of a class which exhibit chaotic behaviour.

Physica 10D (1984) 1–35
North-Holland, Amsterdam

UNIVERSALITY AND COMPLEXITY IN CELLULAR AUTOMATA

Stephen WOLFRAM*
The Institute for Advanced Study, Princeton NJ 08540, USA

Cellular automata are discrete dynamical systems with simple construction but complex self-organizing behaviour. Evidence is presented that all one-dimensional cellular automata fall into four distinct universality classes. Characterizations of the structures generated in these classes are discussed. Three classes exhibit behaviour analogous to limit points, limit cycles and chaotic attractors. The fourth class is probably capable of universal computation, so that properties of its infinite time behaviour are undecidable.

1. Introduction

Cellular automata are mathematical models for complex natural systems containing large numbers of simple identical components with local interactions. They consist of a lattice of sites, each with a finite set of possible values. The value of the sites evolve synchronously in discrete time steps according to identical rules. The value of a particular site is determined by the previous values of a neighbourhood of sites around it.

The behaviour of a simple set of cellular automata were discussed in ref. 1, where extensive references were given. It was shown that despite their simple construction, some cellular automata are capable of complex behaviour. This paper discusses the nature of this complex behaviour, its characterization, and classification. Based on investigation of a large sample of cellular automata, it suggests that many (perhaps all) cellular automata fall into four basic behaviour classes. Cellular automata within each class exhibit qualitatively similar behaviour. The small number of classes implies considerable university in the qualitative behaviour of cellular automata. This universality implies that many details of the construction of a cellular automaton are irrelevant in determining its qualitative behaviour. Thus complex physical and biological systems may lie in the same universality classes as the idealized mathematical models provided by cellular automata. Knowledge of cellular automaton behaviour may then yield rather general results on the behaviour of complex natural systems.

Cellular automata may be considered as discrete dynamical systems. In almost all cases, cellular automaton evolution is irreversible. Trajectories in the configuration space for cellular automata therefore merge with time, and after many time steps, trajectories starting from almost all initial states become concentrated onto "attractors". These attractors typically contain only a very small fraction of possible states. Evolution to attractors from arbitrary initial states allows for "self-organizing" behaviour, in which structure may evolve at large times from structureless initial states. The nature of the attractors determines the form and extent of such structures.

The four classes mentioned above characterize the attractors in cellular automaton evolution. The attractors in classes 1, 2 and 3 are roughly anal-

* Work supported in part by the Office of Naval Research under contract number N00014-80-C0657.

ogous respectively to the limit points, limit cycles and chaotic ("strange") attractors found in continuous dynamical systems. Cellular automata of the fourth class behave in a more complicated manner, and are conjectured to be capable of universal computation, so that their evolution may implement any finite algorithm.

The different classes of cellular automaton behaviour allow different levels of prediction of the outcome of cellular automaton evolution from particular initial states. In the first class, the outcome of the evolution is determined (with probability 1), independent of the initial state. In the second class, the value of a particular site at large times is determined by the initial values of sites in a limited region. In the third class, a particular site value depends on the values of an ever-increasing number of initial sites. Random initial values then lead to chaotic behaviour. Nevertheless, given the necessary set of initial values, it is conjectured that the value of a site in a class 3 cellular automaton may be determined by a simple algorithm. On the other hand, in class 4 cellular automata, a particular site value may depend on many initial site values, and may apparently be determined only by an algorithm equivalent in complexity to explicit simulation of the cellular automaton evolution. For these cellular automata, no effective prediction is possible; their behaviour may be determined only by explicit simulation.

This paper describes some preliminary steps towards a general theory of cellular automaton behaviour. Section 2 below introduces notation and formalism for cellular automata. Section 3 discusses general qualitative features of cellular automaton evolution illustrating the four behaviour classes mentioned above. Section 4 introduces entropies and dimensions which characterize global features of cellular automaton evolution. Successive sections consider each of the four classes of cellular automata in turn. The last section discusses some tentative conclusions.

This paper covers a broad area, and includes many conjectures and tentative results. It is not intended as a rigorous mathematical treatment.

2. Notation and formalism

$a_i^{(t)}$ is taken to denote the value of site i in a one-dimensional cellular automaton at time step t. Each site value is specified as an integer in the range 0 through $k-1$. The site values evolve by iteration of the mapping

$$a_i^{(t)} = \mathbf{F}[a_{i-r}^{(t-1)}, a_{i-r+1}^{(t-1)}, \ldots, a_i^{(t-1)}, \ldots, a_{i+r}^{(t-1)}]. \quad (2.1)$$

F is an arbitrary function which specifies the cellular automaton rule.

The parameter r in eq. (2.1) determines the "range" of the rule: the value of a given site depends on the last values of a neighbourhood of at most $2r+1$ sites. The region affected by a given site grows by at most r sites in each direction at every time step; propagating features generated in cellular automaton evolution may therefore travel at most r sites per time step. After t time steps, a region of at most $1+2rt$ sites may therefore be affected by a given initial site value.

The "elementary" cellular automata considered in ref. 1 have $k=2$ and $r=1$, corresponding to nearest-neighbour interactions.

An alternative form of eq. (2.1) is

$$a_i^{(t)} = \mathbf{f}\left[\sum_{j=-r}^{j=r} \alpha_j a_{i+j}^{(t-1)}\right], \quad (2.2)$$

where the α_j are integer constants, and the function **f** takes a single integer argument. Rules specified according to (2.1) may be reproduced directly by taking $\alpha_j = k^{r-j}$.

The special class of additive cellular automaton rules considered in ref. 2 correspond to the case in which **f** is a linear function of its argument modulo k. Such rules satisfy a special additive superposition principle. This allows the evolution of any initial configuration to be determined by superposition of results obtained with a few basis configurations, and makes possible the algebraic analysis of ref. 2.

"Totalistic" rules defined in ref. 1, and used in several examples below, are obtained by taking

$$\alpha_j = 1 \tag{2.3}$$

in eq. (2.2). Such rules give equal weight to all sites in a neighbourhood, and imply that the value of a site depends only on the total of all preceding neighbourhood site values. The results of section 3 suggest that totalistic rules exhibit behaviour characteristic of all cellular automata.

Cellular automaton rules may be combined by composition. The set of cellular automaton rules is closed under composition, although composition increases the number of sites in the neighbourhood. Composition of a rule with itself yields patterns corresponding to alternate time steps in time evolution according to the rule. Compositions of distinct results do not in general commute. However, if a composition $\mathbf{F}_1\mathbf{F}_2$ of rules generates a sequence of configurations with period π, then the rule $\mathbf{F}_2\mathbf{F}_1$ must also allow a sequence of configurations with period π. As discussed below, this implies that the rules $\mathbf{F}_1\mathbf{F}_2$ and $\mathbf{F}_2\mathbf{F}_1$ must yield behaviour of the same class.

The configuration $a_i = 0$ may be considered as a special "null" configuration ("ground state"). The requirement that this configuration remain invariant under time evolution implies

$$\mathbf{F}[0, 0, \ldots, 0] = 0 \tag{2.4a}$$

and

$$\mathbf{f}[0] = 0 \,. \tag{2.4b}$$

All rules satisfy this requirement if iterated at most k times, at least up to a relabelling of the k possible values.

It is convenient to consider symmetric rules, for which

$$\mathbf{F}[a_{i-r}, \ldots, a_{i+r}] = \mathbf{F}[a_{i+r}, \ldots, a_{i-r}] \,. \tag{2.5}$$

Once a cellular automaton with symmetric rules has evolved to a symmetric state (in which $a_{n+i} = a_{n-i}$ for some n and all i), it may subsequently generate only symmetric states (assuming symmetric boundary conditions), since the operation of space reflection commutes with time evolution in this case.

Rules satisfying the conditions (2.4) and (2.5) will be termed "legal".

The cellular automaton rules (2.1) and (2.2) may be considered as discrete analogues of partial differential equations of order at most $2r + 1$ in space, and first order in time. Cellular automata of higher order in time may be constructed by allowing a particular site value to depend on values of a neighbourhood of sites on a number s of previous time steps. Consideration of "effective" site values $\Sigma_{n=0}^{s-1} m^n a_i^{(t-n)}$ always allows equivalent first-order rules with $k = m^s - 1$ to be constructed.

The form of the function $\mathbf{F}$ in the time evolution rule (2.1) may be specified by a "rule number" [1]

$$\mathbf{R}_\mathbf{F} = \sum_{\{a_{i-r}, a_{i+r}\}} \mathbf{F}[a_{i-r}, \ldots, a_{i+r}] k^{\Sigma_i^r = -r \, k^{r-j} a_{i+j}} \,. \tag{2.6}$$

The function $\mathbf{f}$ in eq. (2.2) may similarly be specified by a numerical "code"

$$\mathbf{C}_\mathbf{f} = \sum_{n=0}^{(2r+1)(k-1)} k^n \mathbf{f}[n] \,. \tag{2.7}$$

The condition (2.4) implies that both $\mathbf{R}_F$ and $\mathbf{C}_f$ are multiples of k.

In general, there are a total of $k^{k^{(2r+1)}}$ possible cellular automaton rules of the form (2.1) or (2.2). Of these, $k^{k^{r+1}(k^r+1)/2-1}$ are legal. The rapid growth of the number of possible rules with r implies that an exponentially small fraction of rules may be obtained by composition of rules with smaller r.

A few cellular automaton rules are "reducible" in the sense that the evolution of sites with particular values, or on a particular grid of positions and times, are independent of other site values. Such cellular automata will usually be excluded from the classification described below.

Very little information on the behaviour of a cellular automaton can be deduced directly from simple properties of its rule. A few simple results are nevertheless clear.

First, necessary (but not sufficient) conditions for a rule to yield unbounded growth are

$$\mathbf{F}[a_{i-r}, a_{i-r+1}, \ldots, a_{i-1}, 0, 0, \ldots, 0] \neq 0\,,$$
$$\mathbf{F}[0, \ldots, 0, 0, a_{i+1}, \ldots, a_{i+r}] \neq 0\,, \qquad (2.8)$$

for some set of a_i. If these conditions are not fulfilled then regions containing nonzero sites surrounded by zero sites can never grow, and the cellular automaton must exhibit behaviour of class 1 or 2. For totalistic rules, the condition (2.8) becomes

$$\mathbf{f}[n] \neq 0 \qquad (2.9)$$

for some $n < r$.

Second, totalistic rules for which

$$\mathbf{f}[n_1] \geq \mathbf{f}[n_2] \qquad (2.10)$$

for all $n_1 > n_2$ exhibit no "growth inhibition" and must therefore similarly be of class 1 or 2.

One may consider cellular automata both finite and infinite in extent.

When finite cellular automata are discussed below, they are taken to consist of N sites arranged around a circle (periodic boundary conditions). Such cellular automata have a finite number k^N of possible states. Their evolution may be represented by finite state transition diagrams (cf. [2]), in which nodes representing each possible configuration are joined by directed arcs, with a single arc leading from a particular node to its successor after evolution for one time step. After a sufficiently long time (less than k^N), any finite cellular automaton must enter a cycle, in which a sequence of configurations is visited repeatedly. These cycles represent attractors for the cellular automaton evolution, and correspond to cycles in the state transition graph. At nodes in the cycles may be rooted trees representing transients. The transients are irreversible in the sense that nodes in the tree have a single successor, but may have several predecessors. In the course of time evolution, all states corresponding to nodes in the trees ultimately evolve through the configurations represented by the roots of the trees to the cycles on which the roots lie. Configurations corresponding to nodes on the periphery of the state transition diagram (terminals or leaves of the transient trees) are never reached in the evolution: they may occur only as initial states. The fraction of configurations which may be reached after one time step in cellular automaton evolution, and which are therefore not on the periphery of the state transition diagram, gives a simple measure of irreversibility.

The configurations of infinite cellular automata are specified by (doubly) infinite sequences of site values. Such sequences are naturally identified as elements of a Cantor set (e.g. [3]). (They differ from real numbers through the inequivalence of configurations such as .111111... and 1.0000...). Cellular automaton rules define mappings from this Cantor set to itself. The mappings are invariant under shifts by virtue of the identical treatment of each site in eqs. (2.1) and (2.2). With natural measures of distance in the Cantor set, the mappings are also continuous. The typical irreversibility of cellular automaton evolution is manifest in the fact that the mapping is usually not injective, as discussed in section 4.

Eqs. (2.1) and (2.2) may be generalized to several dimensions. For $r = 1$, there are at least two possible symmetric forms of neighbourhood, containing $2d + 1$ (type I) and 3^d (type II) sites respectively; for larger r other "unit cells" are possible.

3. Qualitative characterization of cellular automaton behaviour

This section discusses some qualitative features of cellular automaton evolution, and gives empirical evidence for the existence of four basic classes of behaviour in cellular automata. Section 4 introduces some methods for quantitative analysis of cellular automata. Later sections use these methods to suggest fundamental characterizations of the four cellular automaton classes.

Fig. 1 shows the pattern of configurations generated by evolution according to each of the 32 possible legal totalistic rules with $k = 2$ and $r = 2$, starting from a "disordered" initial configuration (in which each site value is independently chosen as 0 or 1 with probability $\frac{1}{2}$). Even with such a structureless initial state, many of the rules are seen to generate patterns with evident structure. While the patterns obtained with different rules all differ in detail, they appear to fall into four qualitative classes:

1) Evolution leads to a homogeneous state (realized for codes 0, 4, 16, 32, 36, 48, 54, 60 and 62).

2) Evolution leads to a set of separated simple stable or periodic structures (codes 8, 24, 40, 56 and 58).

3) Evolution leads to a chaotic pattern (codes 2, 6, 10, 12, 14, 18, 22, 26, 28, 30, 34, 38, 42, 44, 46 and 50).

4) Evolution leads to complex localized structures, sometimes long-lived (codes 20 and 52).

Some patterns (e.g. code 12) assigned to class 3 contain many triangular "clearings" and appear more regular than others (e.g. code 10). The degree of regularity is related to the degree of irreversibility of the rules, as discussed in section 7.

Fig. 2 shows patterns generated from several different initial states according to a few of the cellular automaton rules of fig. 1. Patterns obtained with different initial states are seen to differ in their details, but to exhibit the same characteristic qualitative features. (Expectional initial states giving rise to different behaviour may exist with low or zero probability.) Fig. 3 shows the differences between patterns generated by various cellular automaton rules from initial states differing in the value of a single site.

*This sampling and many other investigations reported in this paper were performed using the C language computer program[4]. Requests for copies of this program should be directed to the author.

Figs. 4, 5 and 6 show examples of various sets of totalistic cellular automata. Fig. 4 shows some $k = 2$, $r = 3$ rules, fig. 5 some $k = 3$, $r = 1$ rules, and fig. 6 some $k = 5$, $r = 1$ rules. The patterns generated are all seen to be qualitatively similar to those of fig. 1, and to lie in the same four classes.

Patterns generated by all possible $k = 2$, $r = 1$ cellular automata were given in ref. 1, and are found to lie in classes 1, 2 and 3. Totalistic $k = 2$, $r = 1$ rules are found to give patterns typical of all $k = 2$, $r = 1$ rules. In general, totalistic rules appear to exhibit no special simplifications, and give rise to behaviour typical of all cellular automaton rules with given k and r.

An extensive sampling of many other cellular automaton rules supports the general conjecture that the four classes introduced above cover all one-dimensional cellular automata*.

Table I gives the fractions of various sets of cellular automata in each of the four classes. With increasing k and r, class 3 becomes overwhelmingly the most common. Classes 1 and 2 are decreasingly common. Class 4 is comparatively rare, but becomes more common for larger k and r.

"Reducible" cellular automata (mentioned in section 2) may generate patterns which contain features from several classes. In a typical case, fixed or propagating "membranes" consisting of sites with a particular value may separate regions containing patterns from classes 3 or 4 formed from sites with other values.

This paper concerns one-dimensional cellular automata. Two-dimensional cellular automata also appear to exhibit a few distinct classes of behaviour. Superficial investigations [5] suggest

Table I
Approximate fractions of legal totalistic cellular automaton rules in each of the four basic classes

Class	$k = 2$, $r = 1$	$k = 2$, $r = 2$	$k = 2$, $r = 3$	$k = 3$, $r = 1$
1	0.50	0.25	0.09	0.12
2	0.25	0.16	0.11	0.19
3	0.25	0.53	0.73	0.60
4	0	0.06	0.06	0.07

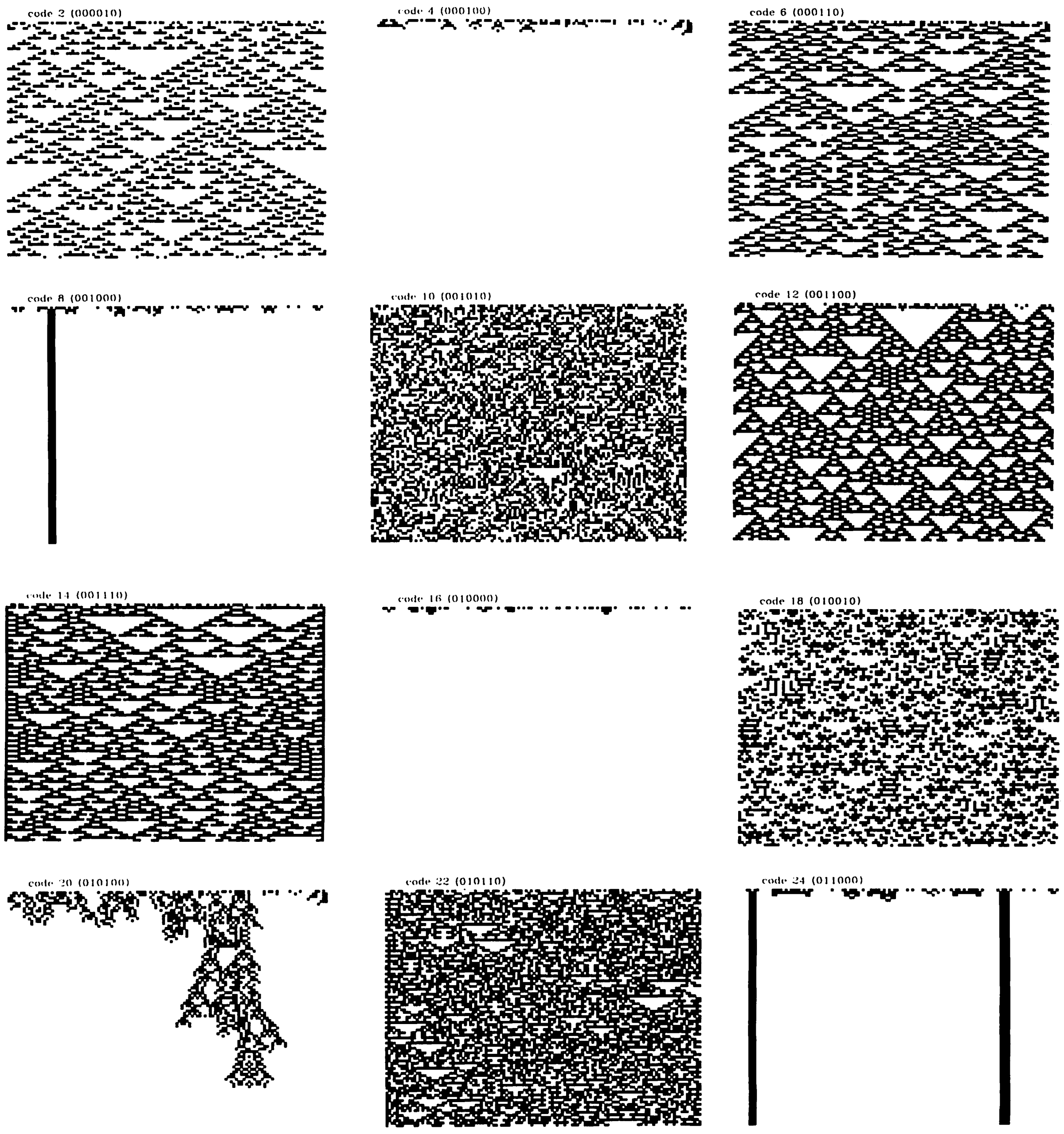

code 2 (000010)
code 4 (000100)
code 6 (000110)
code 8 (001000)
code 10 (001010)
code 12 (001100)
code 14 (001110)
code 16 (010000)
code 18 (010010)
code 20 (010100)
code 22 (010110)
code 24 (011000)

Fig. 1a.

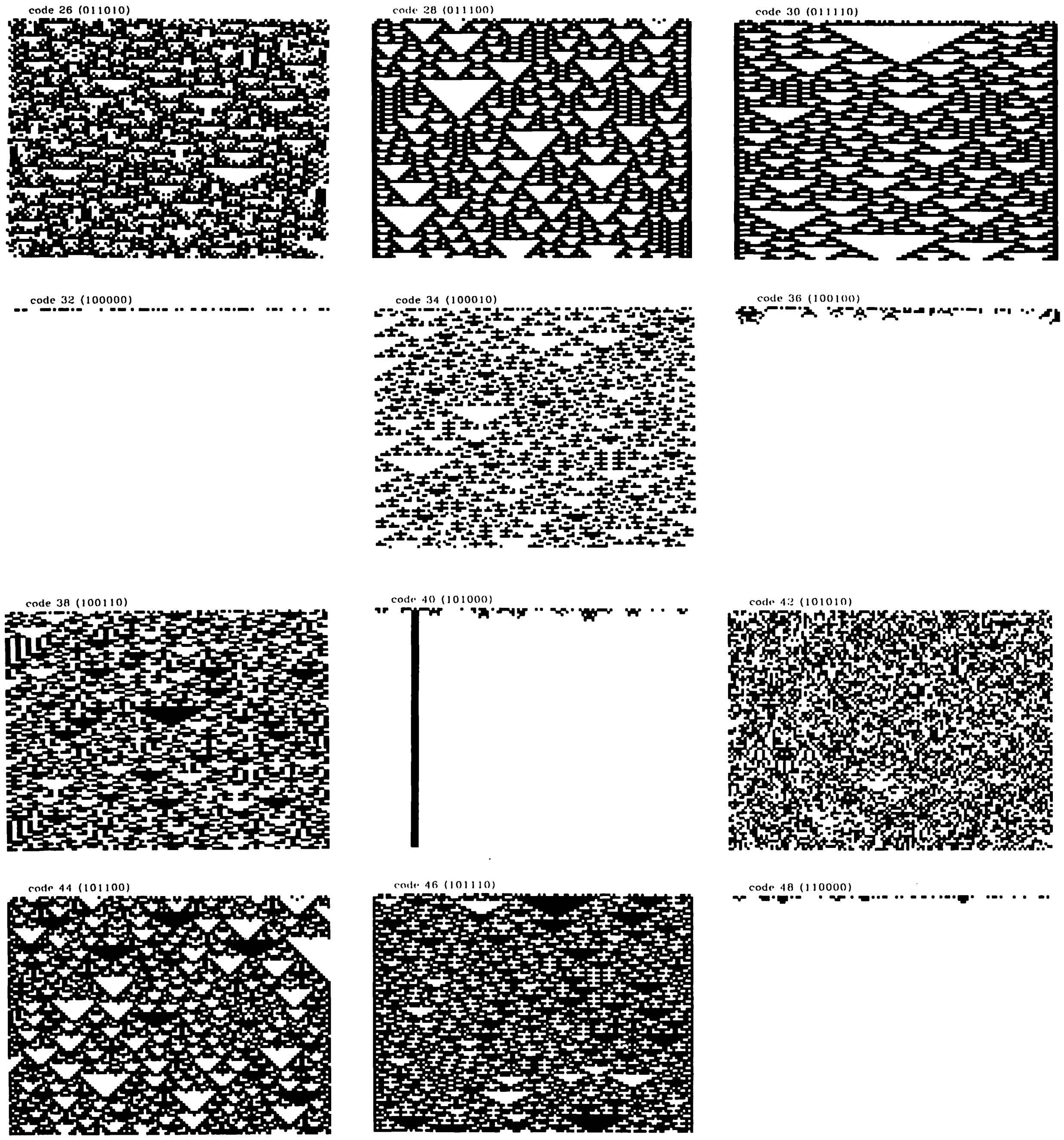

code 26 (011010)
code 28 (011100)
code 30 (011110)
code 32 (100000)
code 34 (100010)
code 36 (100100)
code 38 (100110)
code 40 (101000)
code 42 (101010)
code 44 (101100)
code 46 (101110)
code 48 (110000)

Fig. 1b.

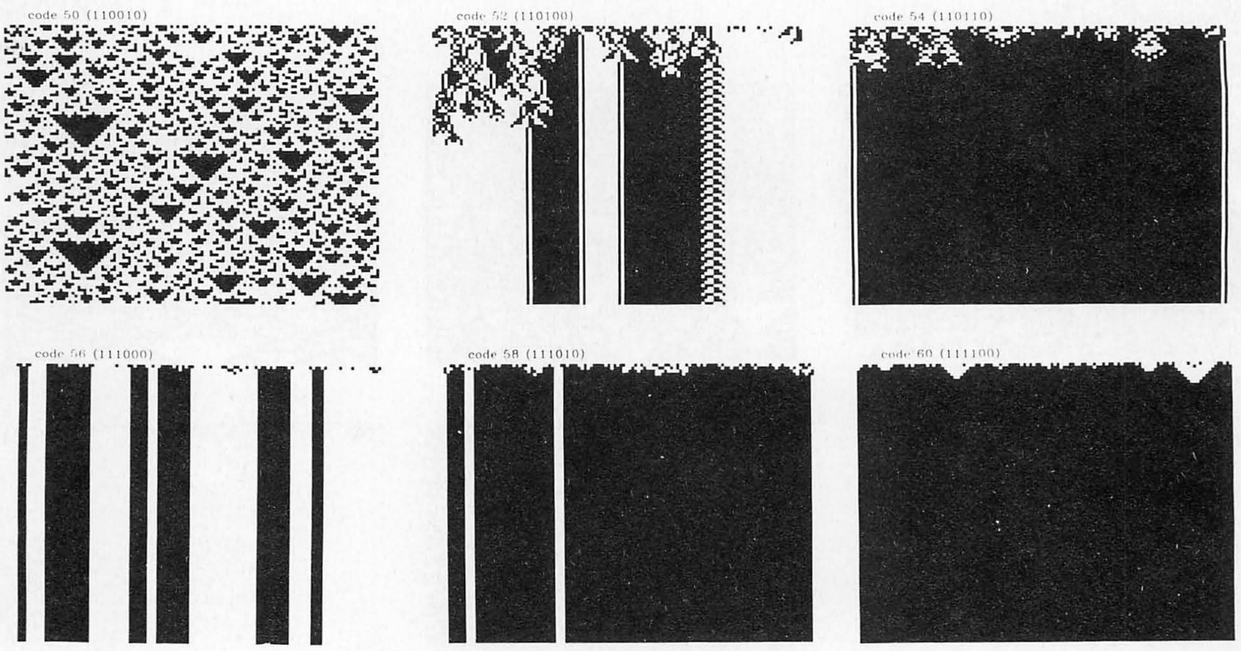


Fig. 1c.

Fig. 1a–c. Evolution of all possible legal one-dimensional totalistic cellular automata with $k = 2$ and $r = 2$. k gives the number of possible values for each site, and r gives the range of the cellular automaton rules. A range $r = 2$ allows the nearest and next-nearest neighbours of a site to affect its value on the next time step. Time evolution for totalistic cellular automata is defined by eqns. (2.2) and (2.7). The initial state is taken disordered, each site having values 0 and 1 with independent equal probabilities. Configurations obtained at successive time steps in the cellular automaton evolution are shown on successive horizontal lines. Black squares represent sites with value 1; white squares sites with value 0. All the cellular automaton rules illustrated are seen to exhibit one of four qualitative classes of behaviour.

that these classes may in fact be identical to the four found in one-dimensional cellular automata.

4. Quantitative characterizations of cellular automaton behaviour

This section describes quantitative statistical measures of order and chaos in patterns generated by cellular automaton evolution. These measures may be used to distinguish the four classes of behaviour identified qualitatively above.

Consider first the statistical properties of configurations generated at a particular time step in cellular automaton evolution. A disordered initial state, in which each site takes on its k possible values with equal independent probabilities, is statistically random. Irreversible cellular automaton evolution generates deviations from statistical randomness. In a random sequence, all k^X possible subsequences ("blocks") of length X must occur with equal probabilities. Deviations from randomness imply unequal probabilities for different subsequences. With probabilities $p_i^{(x)}$ for the k^X possible sequences of site values in a length X block, one may define a specific "spatial set entropy"

$$s^{(x)}(X) = \frac{1}{X}\log_k\left(\sum_{j=1}^{k^X}\theta(p_j^{(x)})\right), \tag{4.1}$$

where $\theta(p) = 1$ for $p > 0$ and $\theta(0) = 0$, and a specific "spatial measure entropy"

$$s_\mu^{(x)}(X) = -\frac{1}{X}\sum_{j=1}^{k^X} p_j^{(x)}\log_k p_j^{(x)}. \tag{4.2}$$

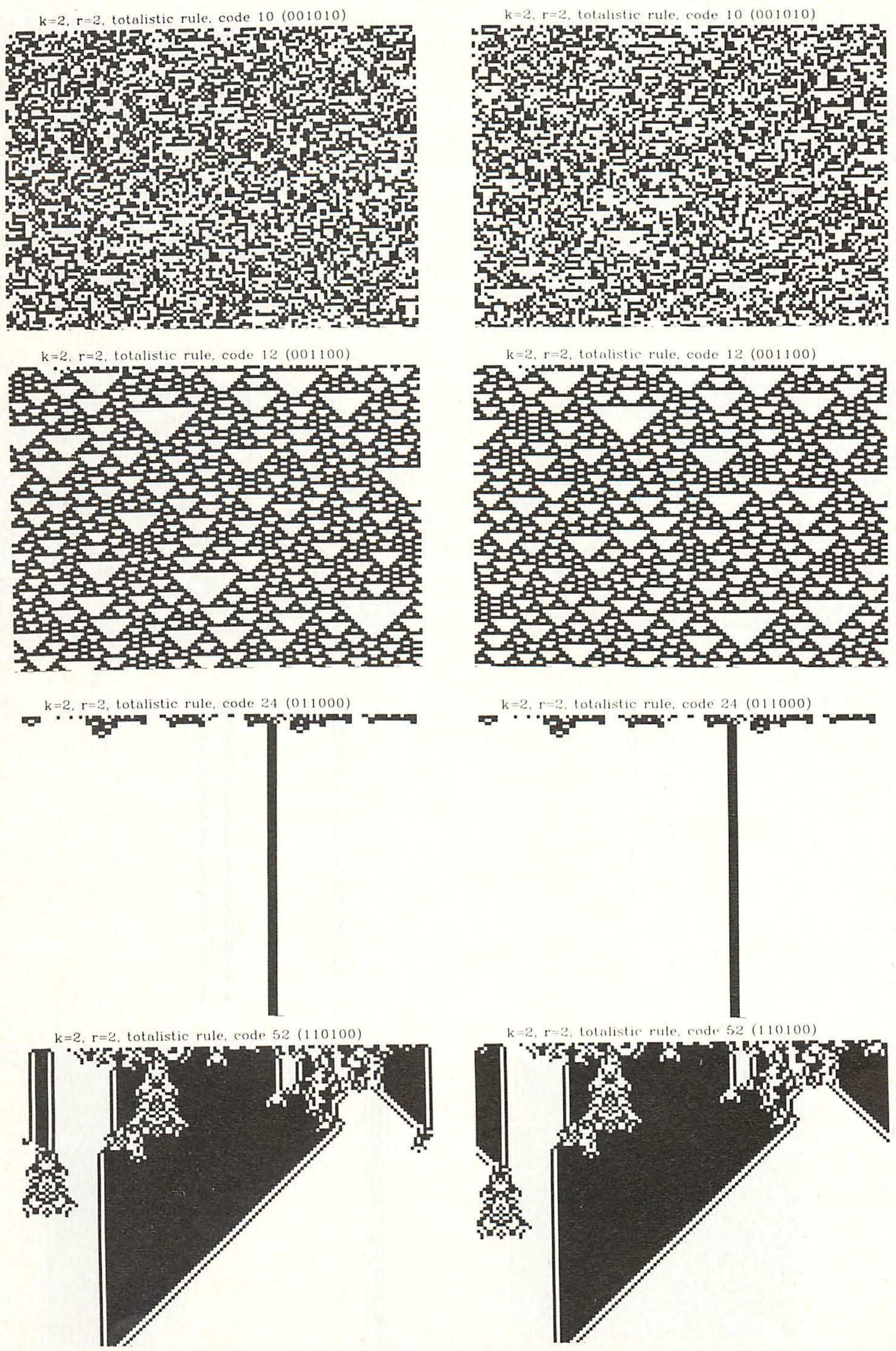

k=2, r=2, totalistic rule, code 10 (001010)
k=2, r=2, totalistic rule, code 10 (001010)
k=2, r=2, totalistic rule, code 12 (001100)
k=2, r=2, totalistic rule, code 12 (001100)
k=2, r=2, totalistic rule, code 24 (011000)
k=2, r=2, totalistic rule, code 24 (011000)
k=2, r=2, totalistic rule, code 52 (110100)
k=2, r=2, totalistic rule, code 52 (110100)

Fig. 2a.

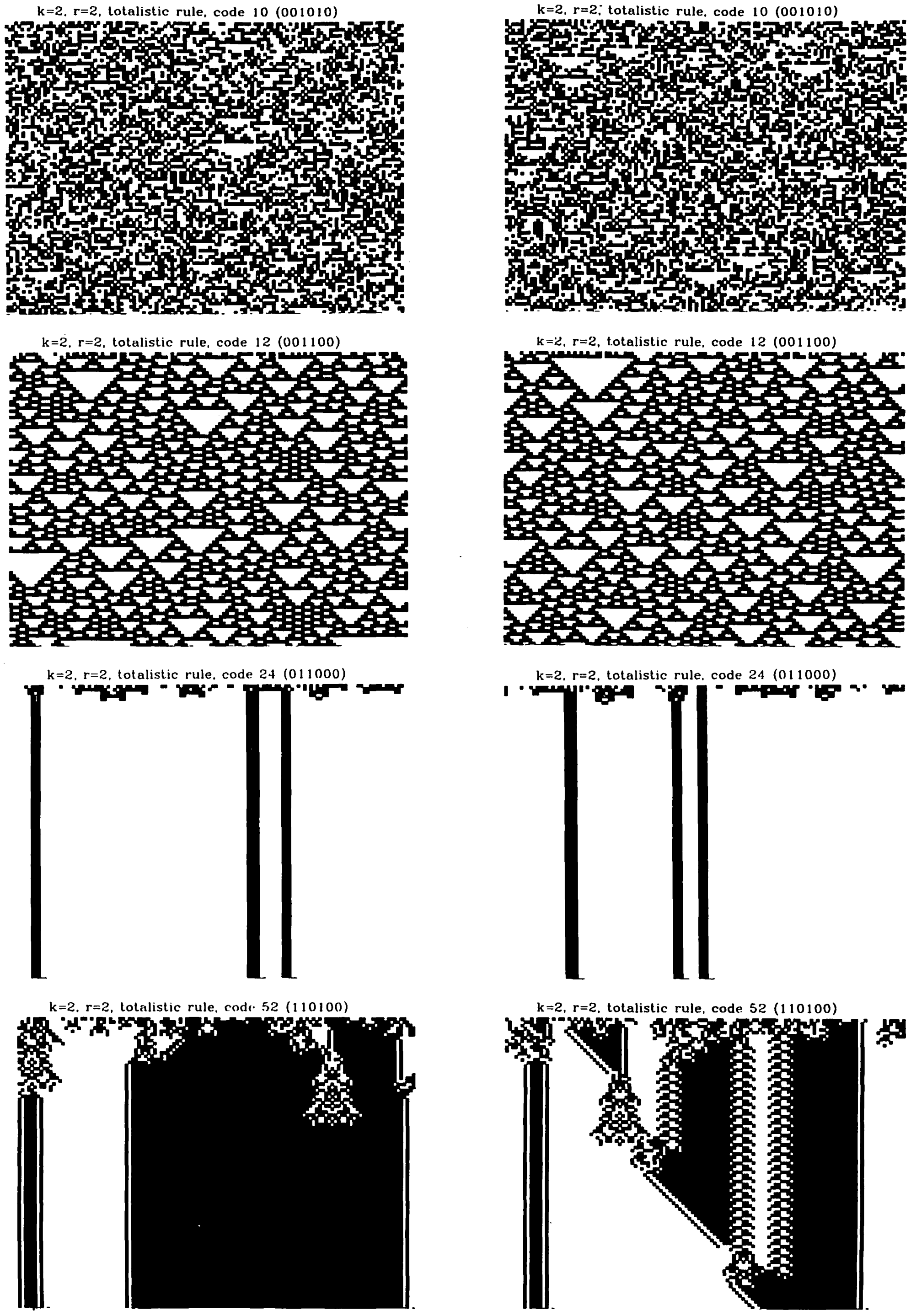

k=2, r=2, totalistic rule, code 10 (001010)
k=2, r=2, totalistic rule, code 10 (001010)
k=2, r=2, totalistic rule, code 12 (001100)
k=2, r=2, totalistic rule, code 12 (001100)
k=2, r=2, totalistic rule, code 24 (011000)
k=2, r=2, totalistic rule, code 24 (011000)
k=2, r=2, totalistic rule, code 52 (110100)
k=2, r=2, totalistic rule, code 52 (110100)

Fig. 2b.

Fig. 2. Evolution of some cellular automata illustrated in fig. 1 from several disordered states. The first two initial states shown differ by a change in the values of two sites, the next by a change in the values of ten sites. The last state is completely different.

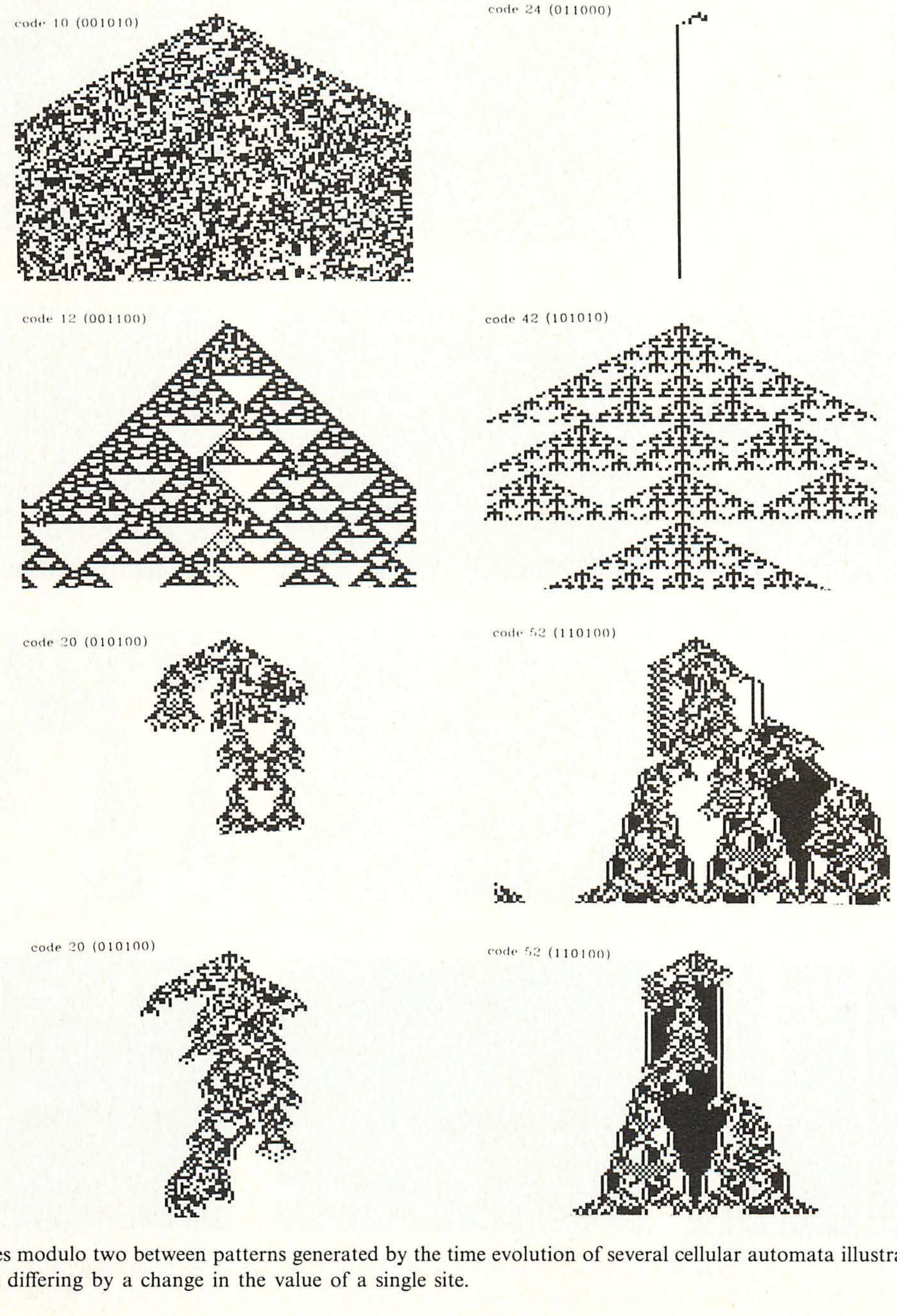


Fig. 3. Differences modulo two between patterns generated by the time evolution of several cellular automata illustrated in fig. 1 with disordered states differing by a change in the value of a single site.

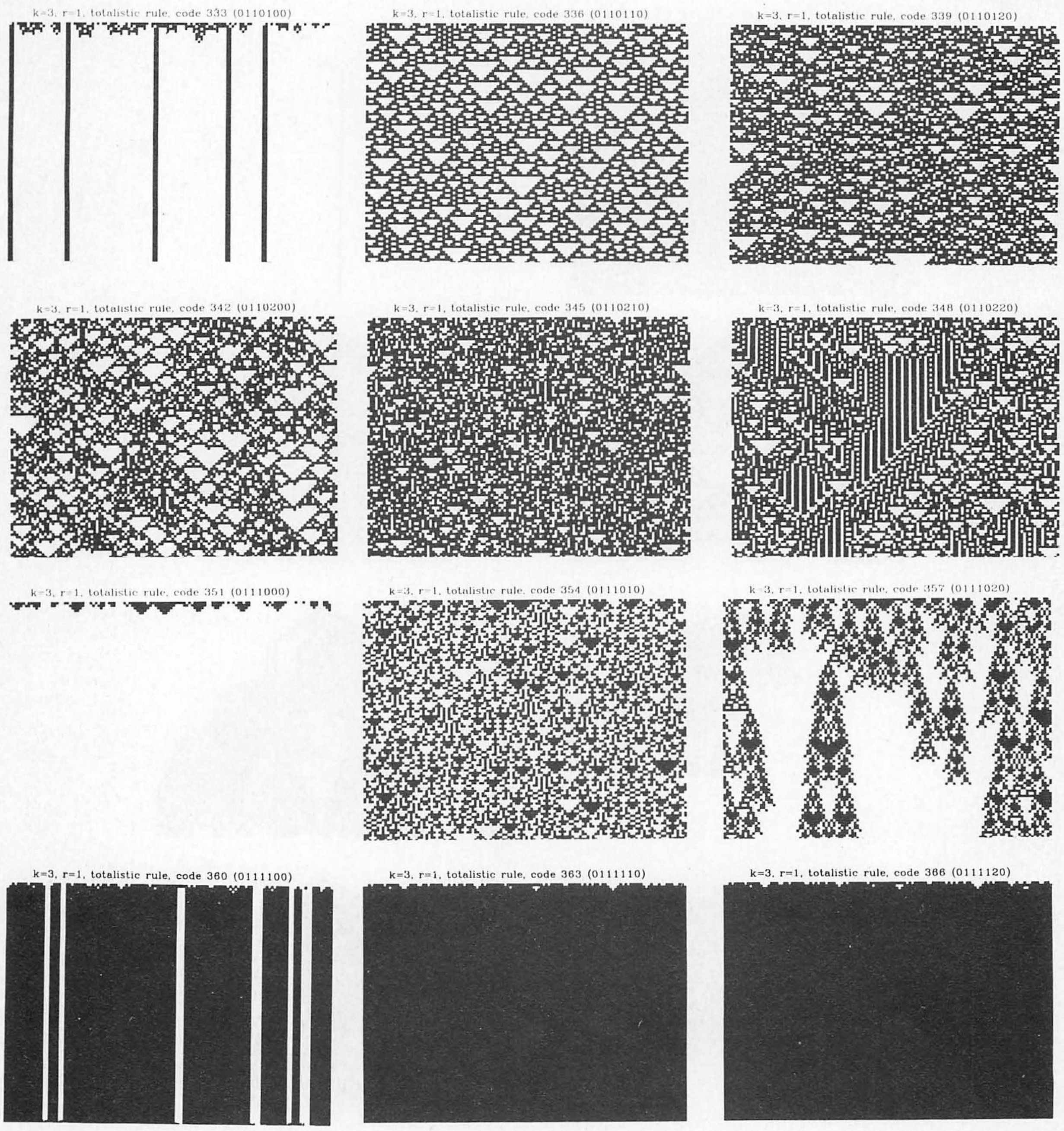
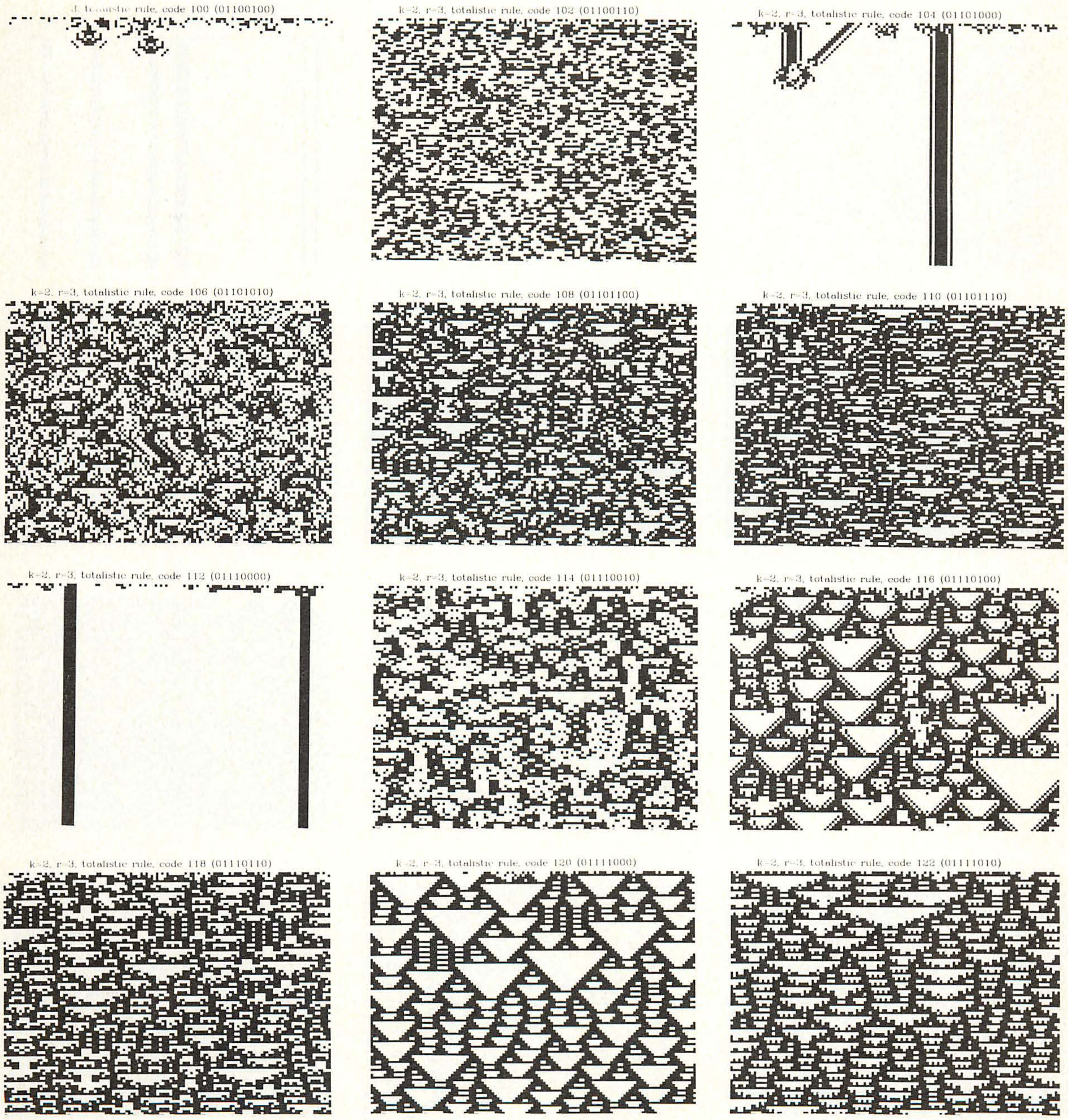

k=3, r=1, totalistic rule, code 333 (0110100)
k=3, r=1, totalistic rule, code 336 (0110110)
k=3, r=1, totalistic rule, code 339 (0110120)
k=3, r=1, totalistic rule, code 342 (0110200)
k=3, r=1, totalistic rule, code 345 (0110210)
k=3, r=1, totalistic rule, code 348 (0110220)
k=3, r=1, totalistic rule, code 351 (0111000)
k=3, r=1, totalistic rule, code 354 (0111010)
k=3, r=1, totalistic rule, code 357 (0111020)
k=3, r=1, totalistic rule, code 360 (0111100)
k=3, r=1, totalistic rule, code 363 (0111110)
k=3, r=1, totalistic rule, code 366 (0111120)

Fig. 4. Examples of the evolution of typical cellular automata with $k = 3$ (three possible site values) and $r = 1$ (only nearest neighbours included in time evolution rules). White squares represent value 0, grey squares value 1, and black squares value 2. The initial state is taken disordered, with each site having values 0, 1 and 2 with equal independent probabilities.



k=2, r=3, totalistic rule, code 100 (01100100)
k=2, r=3, totalistic rule, code 102 (01100110)
k=2, r=3, totalistic rule, code 104 (01101000)
k=2, r=3, totalistic rule, code 106 (01101010)
k=2, r=3, totalistic rule, code 108 (01101100)
k=2, r=3, totalistic rule, code 110 (01101110)
k=2, r=3, totalistic rule, code 112 (01110000)
k=2, r=3, totalistic rule, code 114 (01110010)
k=2, r=3, totalistic rule, code 116 (01110100)
k=2, r=3, totalistic rule, code 118 (01110110)
k=2, r=3, totalistic rule, code 120 (01111000)
k=2, r=3, totalistic rule, code 122 (01111010)

Fig. 5. Examples of the evolution of typical $k = 2$, $r = 3$ cellular automata from a disordered initial state.

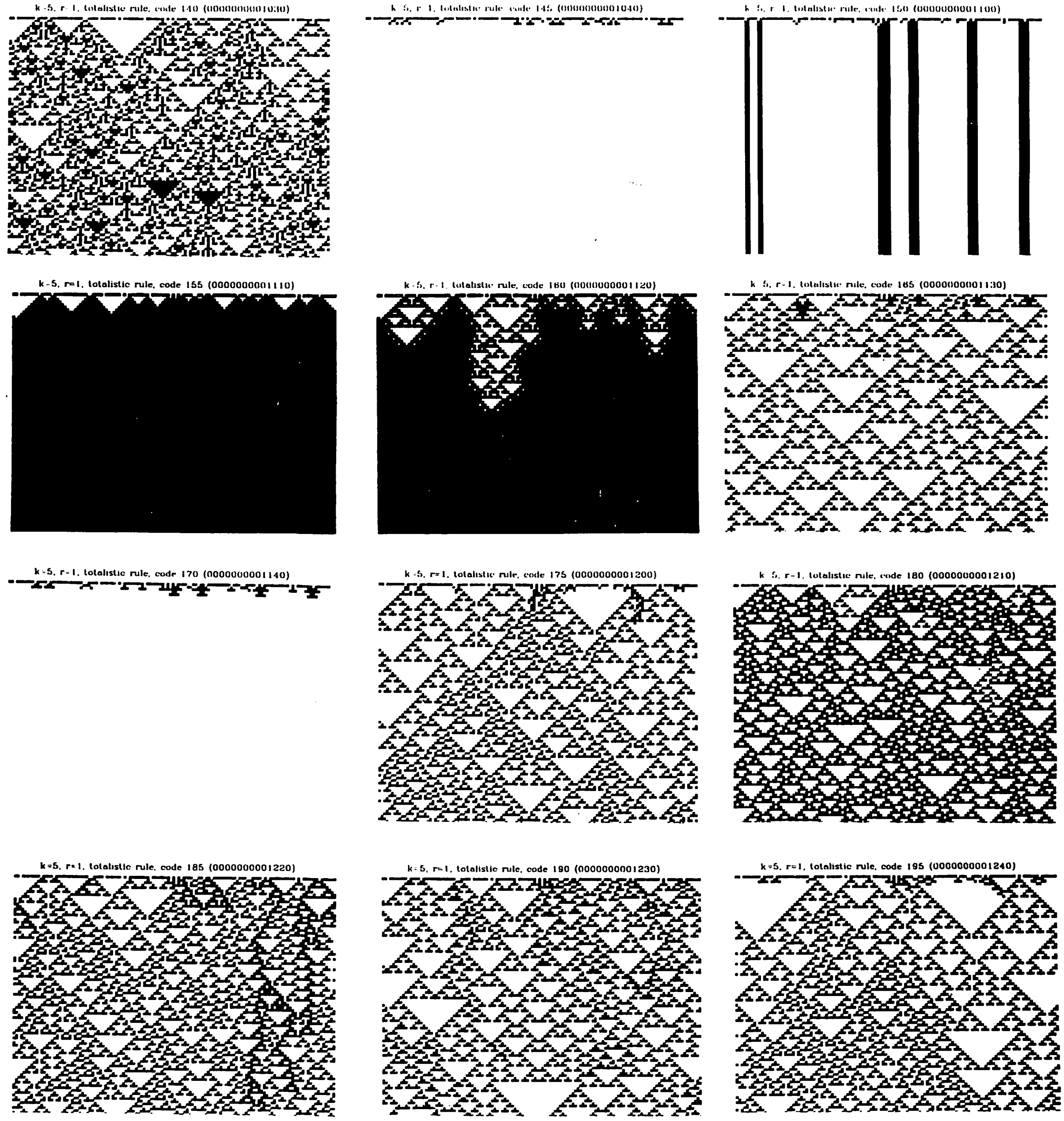

k=5, r=1, totalistic rule, code 140 (0000000001030)
k=5, r=1, totalistic rule, code 145 (0000000001040)
k=5, r=1, totalistic rule, code 150 (0000000001100)
k=5, r=1, totalistic rule, code 155 (0000000001110)
k=5, r=1, totalistic rule, code 160 (0000000001120)
k=5, r=1, totalistic rule, code 165 (0000000001130)
k=5, r=1, totalistic rule, code 170 (0000000001140)
k=5, r=1, totalistic rule, code 175 (0000000001200)
k=5, r=1, totalistic rule, code 180 (0000000001210)
k=5, r=1, totalistic rule, code 185 (0000000001220)
k=5, r=1, totalistic rule, code 190 (0000000001230)
k=5, r=1, totalistic rule, code 195 (0000000001240)

Fig. 6. Examples of the evolution of typical $k = 5$, $r = 1$ cellular automata from a disordered initial state. Darker squares represent sites with larger values.

In both cases, the superscript (x) indicates that "spatial" sequences (obtained at a particular time step) are considered. The "set entropy" (4.1) is determined directly by the total number $N^{(x)}(X)$ of length X blocks generated (with any nonzero probability) in cellular automaton evolution, according to

$$s^{(x)}(X) = \frac{1}{X} \log_k N^{(x)}(X) . \tag{4.3}$$

In the "measure entropy" (4.2) each block is weighted with its probability, so that the result depends explicitly on the probability measure for different cellular automaton configurations, as indicated by the subscript μ. Set entropy is often called "topological entropy"; measure entropy is sometimes referred to as "metric entropy"* (e.g. [6]). For blocks of length 1, the measure entropy $s_\mu^{(x)}(1)$ is related to the densities ρ_i, of sites with each of the k possible values i. $s_\mu^{(x)}(2)$ is related to the densities of "digrams" (blocks of length 2), and so on. In general, the measure entropy gives the average "information content" per site computed by allowing for correlations in blocks of sites up to length X. Note that the entropies (4.1) and (4.2) may be considered to have units of (k-ary) bits per unit distance.

In the equation below, $s_{(\mu)}^{(x)}$ stands for either set entropy $s^{(x)}$ or for measure entropy $s_\mu^{(x)}$.

The definitions (4.1) and (4.2) yield immediately

$$s_\mu^{(x)}(X) \leq s^{(x)}(X) \leq 1 . \tag{4.4}$$

The first inequality is saturated (equality holds) only for "equidistributed" systems, in which all nonzero block probabilities $p_i^{(x)}$ are equal. The second inequality is saturated if all possible length X blocks of site values occur, but perhaps with unequal probabilities. $s_\mu(X) = 1$ only for "X-random" sequences [7], in which all k^X possible sequences of X site values occur with equal probabilities. In addition to (4.4), the definitions (4.1) and (4.2) imply

$$0 \leq s_\mu^{(x)}(X) \leq s^{(x)}(X) . \tag{4.5}$$

$s_\mu^{(x)}(X) = 0$ if and only if just one length X block occurs with nonzero probability, so that $s^{(x)}(X) = 0$ also. As discussed below, the inequality (4.5) is saturated for class 1 cellular automata.

Both set and measure entropies satisfy the subadditivity condition

$$(X_1 + X_2) s_{(\mu)}^{(x)}(X_1 + X_2) \leq X_1 s_{(\mu)}^{(x)}(X_1) + X_2 s_{(\mu)}^{(x)}(X_2) . \tag{4.6}$$

The inequality is saturated if successive blocks of sites are statistically uncorrelated. In general, it implies some decrease in $s_{(\mu)}^{(x)}(X)$ with X (for example, $s_{(\mu)}^{(x)}(2X) \leq s_{(\mu)}^{(x)}(X)$). For cellular automata with translation invariant initial probability measures, stronger constraints may be obtained (analogous to those for "stationary" processes in communication theory [8]). First, note that bounds on $s_{(\mu)}^{(x)}(X)$ valid for any set of probabilities $p_i^{(x)}$ also apply to $s^{(x)}(X)$, since $s^{(x)}(X)$ may formally be reproduced from the definition (4.2) for $s_\mu^{(x)}(X)$ by a suitable (extreme) choice of the $p_i^{(x)}$. The probability $p^{(x)}[a_1, \ldots, a_X]$ for the sequence of site values $a_1, \ldots, a_X$ is given in general by

$$p^{(x)}[a_1, \ldots, a_X] = p^{(x)}[a_1, \ldots, a_{X-1}] p^{(x)}[a_X | a_1, \ldots, a_{X-1}] , \tag{4.7}$$

where $p^{(x)}[a_X | a_1, \ldots, a_{X-1}]$ denotes the conditional probability for a site value a_X, preceded by site values $a_1, \ldots, a_{X-1}$. Defining a total entropy

$$S_\mu^{(x)}[a_1, \ldots, a_X] = -\sum p^{(x)}[a_1, \ldots, a_X] \log_k p^{(x)}[a_1, \ldots, a_X] , \tag{4.8}$$

and corresponding conditional total entropy

*The terms "set" and "measure" entropy, together with "set" and "measure" dimension, are introduced here to rationalize nomenclature.

$$S_{\mu}^{(x)}[a_X|a_1,\dots,a_{X-1}] = -\sum p^{(x)}[a_1,\dots,a_X]\log_k p^{(x)}[a_X|a_1,\dots,a_{X-1}] \leq S_{\mu}^{(x)}[a_1,\dots,a_X]\,, \quad (4.9)$$

one obtains

$$Xs_{\mu}^{(x)}(X) = S_{\mu}^{(x)}(X) \leq \frac{X-1}{X} S_{\mu}^{(x)}(X-1) + \frac{1}{X} S_{\mu}^{(x)}(X)\,. \quad (4.10)$$

Hence,

$$s_{(\mu)}^{(x)}(X) \leq s_{(\mu)}^{(x)}(X-1)\,, \quad (4.11)$$

so that the set and measure entropies for a translationally invariant system decrease monotonically with the block size X. One finds in addition in this case that

$$\Delta_X^2(Xs_{(\mu)}^{(x)}(X)) = (X+1)s_{(\mu)}^{(x)}(X+1) - 2Xs_{(\mu)}^{(x)}(X) + (X-1)s_{(\mu)}^{(x)}(X-1) \leq 0\,, \quad (4.12)$$

so that $Xs_{(\mu)}^{(x)}(X)$ is a convex function of X.

With the definition $s^{(x)}(0)=1$, this implies that there exists a critical block size X_c, such that

$$\begin{aligned} s^{(x)}(X) &= 1, \quad \text{for } X < X_c\,, \\ s^{(x)}(X) &< 1, \quad \text{for } X \geq X_c\,. \end{aligned} \quad (4.13)$$

The significance and values of the critical block size X_c will be discussed in section 7 below.

The entropies $s^{(x)}$ and $s_{\mu}^{(x)}$ may be evaluated either for many blocks in a single cellular automaton configuration, or for blocks in an ensemble of different configurations. For smooth probability measures on the ensemble of possible initial configurations, the results obtained in these two ways are almost always the same. (A probability measure will be considered "smooth" if changes in the values of a few sites in an infinite configuration lead only to infinitesimal changes in the probability for the configuration.) The set entropy $s^{(x)}$ is typically independent of the probability measure on the ensemble, for any smooth measure. The measure entropy $s_{\mu}^{(x)}$ in general depends on the probability measure for initial configurations, although for class 3 cellular automata, it is typically the same for at least a large classes of smooth measures. Notice that with smooth measures, the values of $s^{(x)}(X)$ and $s_{\mu}^{(x)}(X)$ are the same whether the length X blocks used in their computation are taken disjoint or overlapping.

The entropies (4.1) and (4.2) are defined for infinite cellular automata. A corresponding definition may be given for finite cellular automata, with a maximum block length given by the total number of sites N the cellular automaton. The entropies $s^{(x)}(N)$ and $s_{\mu}^{(x)}(N)$ are related to global properties of the state transition diagram for the finite cellular automaton. The value of $s^{(x)}(N)$ at a particular time is determined by the fraction of possible configurations which may be reached at that time by evolution from any initial configuration. The limiting value of $s^{(x)}(N)$ at large times is determined by the fraction of configuration on cycles in the state transition graph. Starting from an initial ensemble in which all kN configurations occur with equal probabilities, the limiting value of $s_{\mu}^{(x)}(N)$ is equal to the limiting value of $s^{(x)}(N)$ if all transient trees in the state transition graph for the finite cellular automaton are identical, so that all configurations with non-zero probabilities are generated with the same probability (cf. [2]).

As mentioned in section 2, the configurations of an infinite cellular automaton may be considered as elements of a Cantor set. For an ensemble of disordered configurations (in which each site takes on its k possible values with equal independent probabilities), this Cantor set has fractal dimension 1. Irreversible cellular automaton evolution may lead to an ensemble of configurations corresponding to elements of a Cantor set with dimension less than one. The limiting value of $s^{(x)}(X)$ as $X\to\infty$ gives the fractal or "set" dimension of this set.

Relations between entropy and dimension may be derived in many ways (e.g. [6, 9]). Consider a set of numbers in the interval [0, 1] of the real line. Divide this interval into k^b bins of width k^{-b}, and let the fraction of bins containing numbers in the set be $N(b)$. For large b (small bin width), this number grows as k^{db}. The exponent d is the Kolmogorov dimension (or "capacity" (cf. [8])) of the set. If the set contains all real numbers in the interval [0, 1], then $N(b) = k^b$, and $d = 1$, as expected. If the set contains only a finite number of points, then $N(b)$ must tend to a constant for large b, yielding $d = 0$. The classic Cantor set consists of real numbers in the interval [0, 1], whose ternary decomposition contains only the digits 0 and 2. Dividing the interval into 3^b equal bins, it is clear that 2^b of these bins contain points in the set. The dimension of the set is thus $\log_3 2$. This dimension may also be found by an explicit recursive geometrical construction, using the fact that the set is "self-similar", in the sense that with appropriate magnification, its parts are identical to the whole.

The example above suggests that one may define a "set dimension" d according to

$$d = \lim_{b\to\infty} \frac{1}{b} \log_k N(b)\,, \tag{4.14}$$

where $N(b)$ is the number of bins which contain elements of the set. The bins are of equal size, and their total number is taken as k^b. Except in particularly pathological examples*, the dimension obtained with this definition is equal to the more usual Hausdorff (or "fractal") dimension (e.g. [11]) obtained by considering the number of patches at arbitrary positions required to cover the set (rather than the number of fixed bins containing elements of the set).

The definition (4.14) may be applied directly to cellular automaton configurations. The k^b "bins" may be taken to consist of cellular automaton configurations in which a block of b sites has a particular sequence of values. The definition (4.3) of set entropy then shows that the set dimension is given by

$$d^{(x)} = \lim_{X\to\infty} s^{(x)}(X)\,. \tag{4.15}$$

A disordered cellular automaton configuration, in which all possible sequences of site values occur with nonzero probability (or an ensemble of such configurations), gives $d^{(x)} = 1$, as expected. Similarly, a homogeneous configuration, such as the null configuration, gives $d^{(x)} = 0$.

The set of configurations which appear at large times in the evolution of a cellular automaton constitute the attractors for the cellular automaton. The set dimension of these attractors is given in terms of the entropies for configurations appearing at large times by eq. (4.15).

Accurate direct evaluation of the set entropy $s^{(x)}(X)$ from cellular automaton configurations typically requires sampling of many more than k^X length X blocks. Inadequate samples yield systematic underestimates of $s^{(x)}(X)$. Direct estimates are most accurate when all nonzero probabilities for length X blocks are equal. In this case, a sample of k^b blocks yields an entropy underestimated on average by approximately

$$\log_k(1 - \exp(-k^{b - Xs(X)}))\,. \tag{4.16}$$

Unequal probabilities increase the magnitude of this error, and typically prevent the generation of satisfactory estimates of $d^{(x)}$ from direct simulations of cellular automaton evolution. (If the probabilities follow a log normal distribution, as in many continuous chaotic dynamical systems [12], then the exponential in eq. (4.16) is apparently replaced by a power [13].)

The dimension (4.15) is given as the limiting exponent with which $N^{(x)}(X)$ increases for large X. In the formula (4.15), this exponent is obtained as the limit of $\log_k[N(X)^{1/X}]$ for large X. If $N^{(x)}(X)$ indeed increases roughly exponentially with X,

* Such as the set formed from the end points of the intervals at each stage in the geometrical construction of the classic Cantor set. This set has zero Hausdorff dimension, but Kolmogorov dimension $\log_3 2$ [9].

then the alternative formula

$$d^{(x)} = \lim_{X\to\infty} \frac{Xs^{(x)}(X)}{(X-1)s^{(x)}(X-1)} = \lim_{X\to\infty} \log_k\left[\frac{N^{(x)}(X)}{N^{(x)}(X-1)}\right] \tag{4.17}$$

is typically more accurate if entropy values are available only for small X.

The set dimension (4.15) may be used to characterize the set of configurations occurring on the attractor for a cellular automaton, without regard to their probabilities. One may also define a "measure dimension" $d_\mu^{(x)}$ which characterizes the probability measure for the configurations (cf. [12]):

$$d_\mu^{(x)} = \lim_{X\to\infty} s_\mu^{(x)}(X)\,. \tag{4.18}$$

It is clear that

$$0 \le d_\mu^{(x)} \le d^{(x)} \le 1\,. \tag{4.19}$$

The measure dimension $d_\mu^{(x)}$ is equal to the "average information per symbol" contained in the sequence of site values in a cellular automaton configuration. If the sequence is completely random (or "∞-random" [7]), then the probabilities $p_i^{(x)}$ for all k^X sequences of length X must be equal for all X, so that $d_\mu^{(x)} = 1$. In this case, there is no redundancy or pattern in the sequence of site values, so that determination of each site value represents acquisition of one (k-ary) bit of information. A cellular automaton configuration with any structure or pattern must give $d_\mu^{(x)} < 1$.

In direct simulations of cellular automaton evolution, the probabilities $p_i^{(x)}$ for each possible length X block are estimated from the frequencies with which the blocks occur. These estimated probabilities are thus subject to Gaussian errors. Although the individual estimated probabilities are unbiased, the measure entropy deduced from them according to eq. (4.2), is systematically biased. Its mean typically yields a systematic underestimate of the true measure entropy, and with fixed sample size, the underestimate deteriorates rapidly with increasing X, making an accurate estimate of $d_\mu^{(x)}$ impossible. However, since an unbiased estimate may be given for any polynomial function of the $p_i^{(x)}$, unbiased estimated upper and lower bounds for the measure entropy may be obtained from estimates for polynomials in $p_i^{(x)}$ just larger and just smaller than $-p_i^{(x)} \log_k p_i^{(x)}$ for $0 \le p_i^{(x)} \le 1$ [14]. In this way, it may be possible to obtain more accurate estimates of $s_\mu^{(x)}$ for large X, and thus of $d_\mu^{(x)}$.

The "spatial" entropies (4.1) and (4.2) were defined in terms of the sequence of site values in a cellular automaton configuration at a particular time step. One may also define "temporal" entropies which characterize the sequence of values taken on by a particular site though many time steps of cellular automaton evolution, as illustrated in fig. 7. With probabilities $p_i^{(t)}$ for the k^T possible sequences of values for a site at T successive time steps, one may define a specific temporal set entropy in analogy with eq. (4.1) by

$$s^{(t)}(T) = \frac{1}{T} \log_k\left(\sum_{j=1}^{k^T} \theta(p_j^{(t)}) \right), \tag{4.20}$$

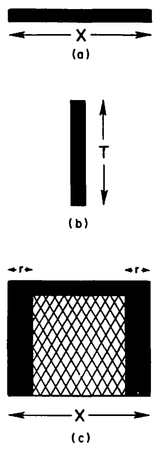


Fig. 7. Space-time regions sampled in the computation of (a) spatial entropies, (b) temporal entropies and (c) patch or mapping entropies. In case (c), the values of sites in the cross-hatched area are completely determined by values in the black "rind".

and a specific temporal measure entropy in analogy with eq. (4.2) by

$$s^{(t)}_{\mu}(T) = -\frac{1}{T}\sum_{j=1}^{k^T} p^{(t)}_j \log_k p^{(t)}_j\,. \tag{4.21}$$

These entropies satisfy relations directly analogous to these given in eqs. (4.3) through (4.6) for spatial entropies. They obey relations analogous to (4.11) and (4.12) only for cellular automata in "equilibrium", statistically independent of time. The temporal entropies (4.20) and (4.21) may be considered to have units of (k-ary) bits per unit time.

Sequences of values in particular cellular automaton configurations typically have little similarity with the "time series" of values attained by a particular site under cellular automaton evolution. The spatial and temporal entropies for a cellular automaton are therefore in general quite different. Notice that the spatial entropy of a cellular automaton configuration may be considered as the temporal entropy of a pure shift mapping applied to the cellular automaton configuration.

Just as dimensions may be assigned to the set of spatial configurations generated in cellular automaton evolution, so also one may assign dimensions to the set of temporal sequences generated by the evolution. The temporal set dimension may be defined in analogy with eq. (4.15) by

$$d^{(t)} = \lim_{T\to\infty} s^{(t)}(T)\,, \tag{4.22}$$

and the temporal measure dimension may be defined by

$$d^{(t)}_{\mu} = \lim_{T\to\infty} s^{(t)}_{\mu}(T)\,. \tag{4.23}$$

If the evolution of a cellular automaton is periodic, so that each site takes on a fixed cycle of values, then

$$d^{(t)} = d^{(t)}_{\mu} = 0\,. \tag{4.24}$$

As discussed in section 6 below, class 2 cellular automata yield periodic structures at large times, so that the correspondingly temporal entropies vanish.

As a generalization of the spatial and temporal entropies introduced above, one may consider entropies associated with space-time "patches" in the patterns generated by cellular automaton evolution, as illustrated in fig. 7. With probabilities $p^{(t,x)}_i$ for the k^{XT} possible patches of spatial width X and temporal extent T, one may define a set entropy

$$s^{(t;x)}(T;X) = \frac{1}{T}\log_k\left(\sum_{j=1}^{k^{XT}} \theta(p^{(t,x)}_j)\right), \tag{4.25}$$

and a measure entropy

$$s^{(t;x)}_{\mu}(T;X) = -\frac{1}{T}\sum_{j=1}^{k^{XT}} p^{(t,x)}_j \log_k p^{(t,x)}_j\,. \tag{4.26}$$

Clearly,

$$s^{(t)}_{(\mu)}(T) = s^{(t;x)}_{(\mu)}(T;1)\,, \tag{4.27}$$
$$s^{(x)}_{(\mu)}(X) = \frac{1}{X} s^{(t;x)}_{(\mu)}(1;X)\,.$$

If no relation existed between configurations at successive time steps then the entropies (4.25) and (4.26) would be bounded simply by

$$s^{(t;x)}_{\mu}(T;X) \le s^{(t;x)}(T;X) \le X\,. \tag{4.28}$$

The cellular automaton rules introduce definite relations between successive configurations and tighten this bound. In fact, the values of all sites in a $T\times X$ space-time patch are determined according to the cellular automaton rules by the values in the "rind" of the patch, as indicated in fig. 7. The rind contains only $X + 2r(T-1)$ sites (where r is the "range" of the cellular automaton rule, defined in section 2), so that

$$s^{(t;x)}_{\mu}(T;X) \le s^{(t;x)}(T;X) \le [X + 2r(T-1)]/T\,. \tag{4.29}$$

For large T (and fixed X), therefore

$$s_{\mu}^{(t;x)}(T;X) \leq s^{(t;x)}(T;X) \leq 2r\,. \tag{4.30}$$

If both X and T tend to infinity with T/X fixed, eq. (4.30) implies that the "information per site" $s_{\mu}^{(t;x)}(T;X)/X$ in a $T \times X$ patch must tend to zero. The evolution of cellular automata can therefore never generate random space-time patterns.

With $T \to \infty$, X fixed, the length X horizontal section of the rind makes a negligible contribution to the entropies. The entropy is maximal if the $2r$ vertical columns in the rind are statistically independent, so that

$$s_{(\mu)}^{(t;x)}(\infty;X) \leq 2rs_{(\mu)}^{(t)}(\infty) = 2rd_{(\mu)}^{(t)}\,. \tag{4.31}$$

In addition,

$$s_{(\mu)}^{(t;x)}(\infty;X) \leq s_{(\mu)}^{(t;x)}(\infty;X+1)\,, \tag{4.32}$$

where the bounds are saturated for large X if the time series associated with different sets of sites are statistically uncorrelated.

The limiting set entropy

$$\mathbf{h} = \lim_{\substack{T\to\infty \\ X\to\infty \\ T/X\to\infty}} s^{(t;x)}(T;X) \tag{4.33}$$

for temporally-extended patches is a fundamental quantity equivalent to the set (or topological) entropy of the cellular automaton mapping in symbolic dynamics. $\mathbf{h}$ may be considered as a dimension for the mapping. It specifies the asymptotic rate at which the number of possible histories for the cellular automaton increases with time. The limiting measure entropy

$$\mathbf{h}_{\mu} = \lim_{\substack{T\to\infty \\ X\to\infty \\ T/X\to\infty}} s_{\mu}^{(t;x)}(T;X) \tag{4.34}$$

gives the average amount of "new information" contained in each cellular automaton configuration, and not already determined from previous configurations. Eqs. (4.31) and (4.32) show that

$$d_{(\mu)}^{(t)} \leq \mathbf{h}_{(\mu)} \leq 2rd_{(\mu)}^{(t)}\,. \tag{4.35}$$

In addition,

$$\mathbf{h}_{(\mu)} \leq 2rd_{(\mu)}^{(x)}\,. \tag{4.36}$$

The basic cellular automaton time evolution rule (2.1) implies that the value a_i of a site i at a particular time step depends on sites a maximum distance r away on the previous time step according to the function $\mathbf{F}[a_{i-r}, \ldots, a_{i+r}]$. After T time steps, the values of the site could depend on sites at distances up to rT, so that features in patterns generated by cellular automaton evolution could propagate at "speeds" up to r sites per time step. For many rules, however, the value of a site after many time steps depends on fewer initial site values, and features may propagate only at lower speeds. In general, let $\|\mathbf{F}^T\|$ denote the minimum R for which the value of site i depends only on the initial values of sites $i-R, \ldots, i+R$. Then the maximum propagation speed associated with the cellular automaton rule $\mathbf{F}$ may be defined as

$$\lambda_{+} = \overline{\lim_{T\to\infty}}\, \|\mathbf{F}^T\|/T\,. \tag{4.37}$$

(The rule is assumed symmetric; for nonsymmetric rules, distinct left and right propagation speeds may be defined.) Clearly,

$$\lambda_{+} \leq r\,. \tag{4.38}$$

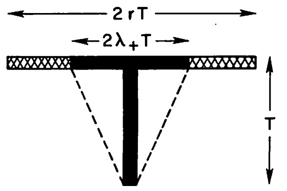


Fig. 8. Pattern of dependence of temporal sequences on spatial sequences, used in the proof of inequalities between spatial and temporal entropies.

When $\lambda_+ = 0$, finite regions of the cellular automaton must ultimately become isolated, so that

$$d^{(t)}_{(\mu)} = \mathbf{h}^{(t)}_{(\mu)} = 0 . \tag{4.39}$$

The construction of fig. 8 shows that for any T,

$$s^{(t)}_{(\mu)}(T) \leq 2rs^{(x)}_{(\mu)}(2rT) . \tag{4.40}$$

In the limit $T \to \infty$, the construction implies

$$d^{(t)}_{(\mu)} \leq 2\lambda_+ d^{(x)}_{(\mu)} , \tag{4.41}$$

The ratio of temporal to spatial entropy is thus bounded by the maximum propagation speed in the cellular automaton. The relation is consistent with the assignment of units to the spatial and temporal entropies mentioned above.

The corresponding inequalities for mapping entropies are:

$$\begin{aligned} &d^{(t)}_{(\mu)} \leq \mathbf{h}_{(\mu)} \leq 2\lambda_+ d^{(x)}_{(\mu)} , \\ &\mathbf{h}_{(\mu)} \leq 2rd^{(t)}_{(\mu)} . \end{aligned} \tag{4.42}$$

The quantity λ_+ defined by eq. (4.37) gives the maximum speed with which any feature in a cellular automaton may propagate. With many cellular automaton rules, however, almost all "features" propagate much more slowly. To define an appropriate maximum average propagation speed, consider the effect after many time steps of changes in the initial state. Let $G(|x - x'|; t)$ denote the probability that the value of a site at position x' is changed when the value of a site at position x is changed t time steps before. The form of $G(|x - x'|; t)$ for various cellular automaton rules is suggested by fig. 3. $G(|x - x'|; t)$ may be considered as a Green function for the cellular automaton evolution. For large t, $G(|x - x'|; t)$ typically vanishes outside a "cone" defined by $|x - x'| = \bar{\lambda}_+ t$. $\bar{\lambda}_+$ may then be considered as a maximum average propagation speed. In analogy with eqs. (4.41) and (4.42), one expects

$$d^{(t)}_{(\mu)} \leq \mathbf{h}_{(\mu)} \leq 2\bar{\lambda}_+ d^{(t)}_{(\mu)} . \tag{4.43}$$

Mapping and temporal entropies thus vanish for cellular automata with zero maximum average propagation speed. Cellular automata in class 2 have this property.

The maximum average propagation speed $\bar{\lambda}_+$ specifies a cone outside which $G(|x - x'|; t)$ almost always vanishes. One may also define a minimum average propagation speed $\bar{\lambda}_-$, such that $G(|x - x'|; t) > 0$ for almost any $|x - x'| < \bar{\lambda}_-$.

The Green function $G(|x - x'|; t)$ gives the probability that a particular site is affected by changes in a previous configuration. The total effect of changes may be measured by the "Hamming distance" $H(t)$ between configurations before and after the changes, defined as the total number of site values which differ between the configurations after t time steps. ($H(t)$ is analogous to Lyapunov exponents for continuous dynamical systems.) Changing the values of initial sites in a small region, $H(t)$ may be given as a space integral of the Green function, and for large t obeys the inequality

$$H(t)/t \leq 2\bar{\lambda}_+ , \tag{4.44}$$

to be compared with the result (4.43) obtained above.

The definitions and properties of dimension given above suggests that the behaviour these quantities determines the degree of "chaotic" behaviour associated with cellular automaton evolution. "Spatial chaos" occurs when $d^{(x)}_{(\mu)} > 0$, and "temporal chaos" when $d^{(t)}_{(\mu)} > 0$. Temporal chaos requires a nonzero maximum average propagation speed for features in cellular automaton patterns, and implies that small changes in initial conditions lead to effects ever-increasing with time.

5. Class 1 cellular automata

Class 1 cellular automata evolve after a finite number of time steps from almost all initial states to a unique homogeneous state, in which all sites have the same value. Such cellular automata may be considered to evolve to simple "limit points" in phase space; their evolution completely destroys any information on the initial state. The spatial and temporal dimensions for such attractors are zero.

Rules for class 1 cellular automata typically take the function **F** of eq. (2.1) to have the same value for almost all of its $k^{(2r+1)}$ possible sets of arguments.

Some exceptional configurations in finite class 1 cellular automata may not evolve to a homogeneous state, but may in fact enter non-trivial cycles. The fraction of such exceptional configurations appears to decrease very rapidly with the size N, suggesting that for infinite class 1 cellular automata the set of exceptional configurations is always of measure zero in the set of all possible configurations. For (legal) class 1 cellular automata whose usual final state has $a_i = n$, $n \neq 0$ (such as code 60 in fig. 1), the null configuration is exceptional for any size N, and yields $a_i = 0$.

6. Class 2 cellular automata

Class 2 cellular automata serve as "filters" which generate separated simple structures from particular (typically short) initial site value sequences*. The density of appropriate sequences in a particular initial state therefore determines the statistical properties of the final state into which it evolves. (There is therefore no unique large-time (invariant) probability measure on the set of possible configurations.) Changes of site values in the initial state almost always affect final site values only within a finite range, typically of order r. The maximum average propagation speed $\bar{\lambda}_+$ defined in section 4 thus vanishes for class 2 cellular automata. The temporal and mapping (but not spatial) dimensions for such automata therefore also vanish.

*They are thus of direct significance for digital image processing.

Although $\bar{\lambda} = 0$ for all class 2 cellular automata, λ is often nonzero. Thus exceptional initial state may exist, from which, for example, unbounded growth may occur. Such initial states apparently occur with probability zero for ensembles of (spatially infinite) cellular automata with smooth probability measures.

The simple structures generated by class 2 cellular automata are either stable, or are periodic, typically with small periods. The class 2 rules with codes 8, 24, 40 and 56 illustrated in fig. 1 all apparently exhibit only stable perisistent structures. Examples of class 2 cellular automata which yield periodic, rather than stable, persistent structures include the $k = 2$, $r = 1$ cellular automaton with rule number 108 [1], and the $k = 3$, $r = 1$ totalistic cellular automaton with code 198. The periods of persistent structures generated in the evolution of class 2 cellular automata are usually less than $k!$. However, examples have been found with larger periods. One is the $k = 2$, $r = 3$ totalistic cellular automata with code 228, in which a persistent structure with period 3 is generated.

The finiteness of the periods obtained at large times in class 2 cellular automata implies that such systems have $d^{(t)}_{(\mu)} = \mathbf{h}_{(\mu)} = 0$, as deduced above from the vanishing of $\bar{\lambda}_+$. The evolution of class 2 cellular automata to zero (temporal) dimension attractors is analogous to the evolution of some continuous dynamical systems to limit cycles.

The set of persistent structures generated by a given class 2 cellular automaton is typically quite simple. For some rules, there are only a finite number of persistent structures. For example, for the code 8 and code 40 rules of fig. 1, only the sequence 111 (surrounded by 0 sites) appears to be persistent. For code 24, 111 and 1111 are both persistent. Other rules yield an infinite sequence of peristent structures, typically constructed by a simple process. For example, with code 56 in fig. 1, any sequence of two or more consecutive 1 sites is persistent.

In general, it appears that the set of persistent structures generated by any class 2 cellular automaton corresponds to the set of words generated by a regular grammar. A regular grammar [15–18] (or "sofic system" [19]) specifies a regular language, whose legal works may be recognized by a finite automaton, represented by a finite state transition graph. A sequence of symbols (site values) specifies a particular traversal of the state transition graph. The traversal begins at a special "start" node; the symbol sequence represents a legal word only if the traversal does not end at an absorbing "stop" node. Each successive symbol in the sequence causes the automaton to make a transition from one state (node) to one of k others, as specified by the state transition graph. At each step, the next state of the automaton depends only on its current state, and the current symbol read, but not on its previous history.

The set of configurations (symbol sequences) generated from all possible initial configurations by one time step of cellular automaton evolution may always be specified by a regular grammar. To determine whether a particular configuration $a^{(1)}$ may be generated after one time step of cellular automaton evolution, one may attempt to construct an explicit predecessor $a^{(0)}$ for it. Assume that a predecessor configuration has been found which reproduces all site values up to position i. Definite values $a_j^{(0)}$ for all $j \leq i - r$ are then determined. Several of the total of k^{2r} sequences of values $a_{i-r+1}^{(0)}, \ldots, a_{i+r+1}^{(0)}$ may be possible. Each sequence may be specified by an integer $q = \Sigma_{j=0}^{2r} k^j a_{i-r+j+1}^{(0)}$. An integer ψ_i between 0 and $2^{k^{2r}}$ may then be defined, with the qth binary bit in ψ_i equal to one if sequence q is allowed, and 0 otherwise. Each possible value of ψ may be considered to correspond to a state in a finite automaton. $\psi = 0$ corresponds to a "stop" state, which is reached if and only if $a^{(1)}$ has no predecessors. Possible values for $a_{i+r+1}^{(0)}$ are then found from ψ_i and the value of $a_{i+1}^{(0)}$. These possible values then determine the value of ψ_{i+1}. A finite state transition graph, determined by the cellular automaton rules, gives the possible transitions $\psi_i \rightarrow \psi_{i+1}$. Configurations reached after one time step of cellular automaton evolution may thus be recognized by a finite automaton with at most $2^{k^{2r}}$ states. The set of such configurations is thus specified by a regular grammar.

In general, if the value of a given site after t steps of cellular automaton evolution depends on m initial site values, then the set of configurations generated by this evolution may be recognized by a finite automaton with at most 2^{k^m} states. The value of m may increase as $2rt$, potentially requiring an infinite number of states in the recognizing automaton, and preventing the specification of the set of possible configurations by a regular grammar. However, as discussed above, the value of m for a class 2 cellular automaton apparently remains finite as $t \rightarrow \infty$. Thus the set of configurations which may persist in such a cellular automaton may be recognized by a finite automaton, and are therefore specified by a regular grammar. The complexity of this grammar (measured by the minimum number of states required in the state transition graph for the recognizing automaton) may be used to characterize the complexity of the large time behaviour of the cellular automaton.

Finite class 2 cellular automata usually evolve to short period cycles containing the same persistent structures as are found in the infinite case. The fraction of exceptional initial states yielding other structures decreases rapidly to zero as N increases.

7. Class 3 cellular automata

Evolution of infinite class 3 cellular automata from almost all possible initial states leads to aperiodic ("chaotic") patterns. After sufficiently many time steps, the statistical properties of these patterns are typically the same for almost all initial states. In particular, the density of nonzero sites typically tends to a fixed nonzero value (often close to $1/k$). In infinite cellular automata, "equilibrium" values of statistical quantities are approached roughly exponentially with time, and are typically attained to high accuracy after a very few time steps. For a few rules (such as the $k = 2$, $r = 1$ rule with rule number 18 [20]), however,

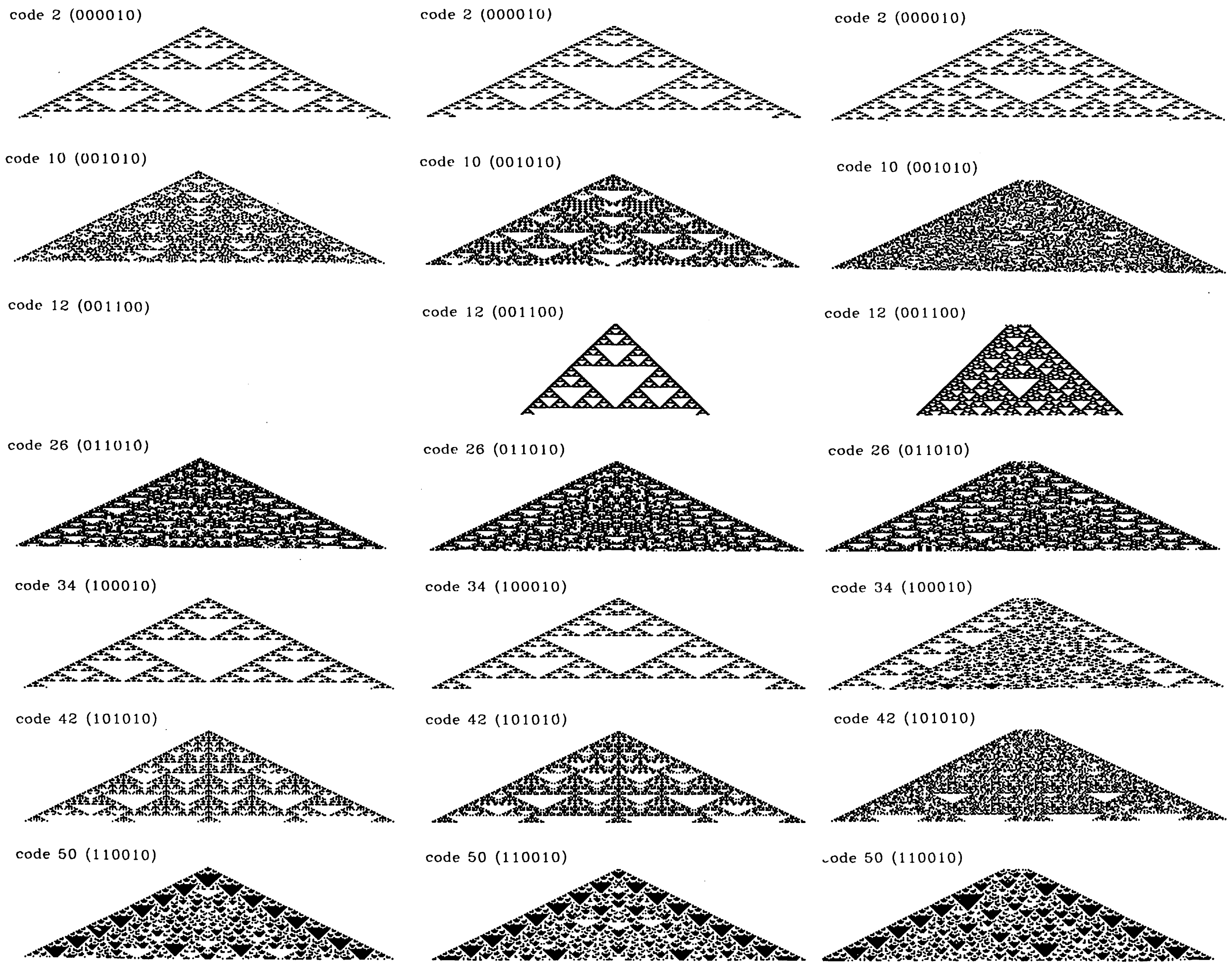


Fig. 9. Evolution of some class 3 totalistic cellular automata with $k = 2$ and $r = 2$ (as illustrated in fig. 1) from initial states containing one or a few nonzero sites. Some cases yield asymptotically self-similar patterns, while others are seen to give irregular patterns.

"defects" consisting of small groups of sites may exist, and may execute approximate random walks, until annihilating, usually in pairs. Such processes lead to transients which decrease with time only as $t^{-1/2}$.

Fig. 1 showed examples of the patterns generated by evolution of some typical class 3 cellular automata from disordered initial states. The patterns range from highly irregular (as for code 10), to rather regular (as for code 12). The most obvious regularity is the appearance of large triangular "clearings" in which all sites have the same value. These clearings occur when a "fluctuation" in which a sequence of consequence of consecutive sites have the same value, is progressively destroyed by the effects of other sites. The rate at which "information" from other sites may "flow" into the fluctuation, and thus the slope of the boundaries of the clearing, may range from $1/k$ to r sites per time step. The qualitative regularity of patterns generated by some class 3 rules arises from the high density of long sequences of correlated site values, and thus of triangular clearings. In general, however, it appears that the density of clearings decreases with their size n roughly as σ^{-n}. Different cellular automata appear to yield a continuous range of σ values. Those with larger σ yield more regular patterns, while those with smaller σ yield more irregular patterns. No sharp distinction appears to exist between class 3 cellular automata yielding regular and irregular patterns.

The first column in fig. 9 shows patterns obtained by evolution with typical class 3 cellular automaton rules from initial states containing a single nonzero site. Unbounded growth, leading to an asymptotically infinite number of nonzero sites, is evident in all cases. Some rules are seen to give highly regular patterns, others lead to irregular patterns.

The regular patterns obtained with rules such as code 2 are asymptotically self-similar fractal curves (cf. [11]). Their form is identical when viewed at different magnifications, down to length scales of order r sites. The total number of nonzero sites in such patterns after t time steps approaches t^d; where d gives the fractal dimension of the pattern. Many class 3 $k = 2$ rules generate a similar pattern, illustrated by codes 2 and 34 in fig. 9, with $d = \log_2 3 \approx 1.59$. Some rules yield self-similar patterns with other fractal dimensions (for example, code 38 yields $d \approx 1.75$), but all self-similar patterns have $d < 2$, and lead to an asymptotic density of sites which tends to zero as t^{d-2}.

Rule such as code 10 are seen to generate irregular patterns by evolution even from a single site initial state. The density of nonzero sites in such patterns is found to tend asymptotically to a nonzero value; in some, but not all, cases the value is the same as would be obtained by evolution from a disordered initial state. The patterns appear to exhibit no large-scale structure.

Cellular automata contain no intrinsic scale beyond the size of neighbourhood which appears in their rules. A configuration containing a single nonzero site is also scale invariant, and any pattern obtained by evolution from it with cellular automaton rules must be scale invariant. The regular patterns in fig. 9 achieve this scale invariance by their self-similarity. The irregular patterns presumably exhibit correlations only over a finite range, and are therefore effectively uniform and scale invariant at large distances.

The second and third columns in fig. 11 shows the evolution of several typical class 3 cellular automata from initial states with nonzero sites in a small region. In some cases (such as code 12), the regular fractal patterns obtained with single nonzero sites are stable under addition of further nonzero initial sites. In other cases (such as code 2) they are seen to be unstable. The numbers of rules yielding stable and unstable fractal patterns are found to be roughly comparable.

Many but not all rules which evolve to regular fractal patterns from simple initial states generate more regular patterns in evolution from disordered initial states. Similarly, many but not all rules which produce stable fractal patterns yield more regular patterns from disordered initial states. For example, code 42 in figs. 1 and 9 generates stable fractal patterns from simple initial state, but

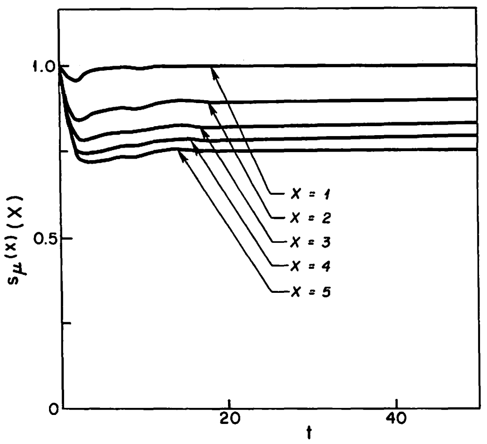
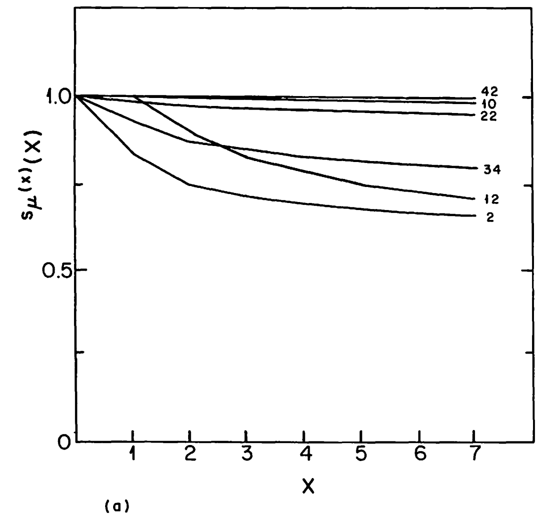


Fig. 10. Evolution of spatial measure entropies $s_\mu^{(x)}(X)$ as a function of time for evolution of the class 3 cellular automaton with code 12 illustrated in fig. 1 from a disordered initial state. The irreversibility of cellular automaton evolution results in a decrease of the entropies with time. Rapid relaxation to an "equilibrium" state is nevertheless seen.

leads to an irregular patterns under evolution from a disordered state. (Although not necessary for such behaviour, this rule possesses the additivity property mentioned in section 2.)

The methods of section 4 may be used to analyse the general behaviour of class 3 cellular automata evolving from typical initial states, in which all sites have nonzero values with nonzero probability. Class 3 cellular automata apparently always exhibit a nonzero minimum average propagation speed $\bar{\lambda}_-$. Small changes in initial states thus almost always lead to increasingly large changes in later states. This suggests that both spatial and temporal dimensions $d_{(\mu)}^{(x)}$ and $d_{(\mu)}^{(t)}$ should be nonzero for all class 3 cellular automata. These dimensions are determined according to eqs. (4.15), (4.18), (4.22) and (4.23) by the limiting values of spatial and temporal entropies.

A disordered or statistically random initial state, in which each site takes on its k possible values with equal independent probabilities, has maximal spatial entropy $s_{(\mu)}^{(x)}(X) = 1$ for all block lengths X. Fig. 10 shows the behaviour of $s_\mu^{(x)}(X)$ as a function of time for several block lengths X in the evolution of a typical class 3 cellular automaton from a disordered (maximal entropy) initial state. The entropies are seen to decrease for a few time steps, and then to reach "equilibrium" values. The "equilibrium" values of $s_\mu^{(x)}(X)$ for class 3 cellular automata are typically independent of the probability measure on the ensemble of possible initial states, at least for "smooth" measures. The decrease in



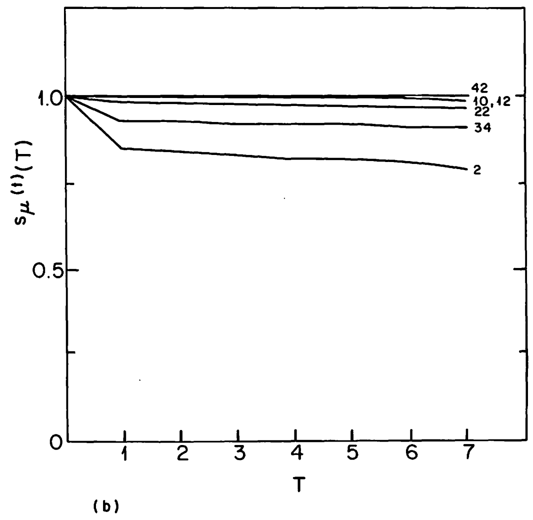


Fig. 11. Evolution of (a) spatial and (b) temporal measure entropies $s_\mu^{(x)}(X)$ and $s_\mu^{(t)}(T)$ obtained at equilibrium by evolution of several class 3 cellular automata illustrated in fig. 1, as a function of the spatial and temporal block lengths X and T. The entropies are evaluated for the region indicated in figs. 7(a) and 7(b). The limit of $s_\mu^{(x)}(X)$ as $X \to \infty$ is the spatial measure dimension of the attractor for the system; the limit of $s_\mu^{(t)}(T)$ as $T \to \infty$ is the temporal measure dimension.

entropy with time manifests the irreversible nature of the cellular automaton evolution. The decrease is found to be much greater for class 3 cellular automata which generate regular patterns (with many triangular clearings) than for those which yield irregular patterns. The more regular patterns require a higher degree of self-organization, with correspondingly greater irreversibility, and larger entropy decrease.

As discussed in section 4, the dependence of $s^{(x)}_{(\mu)}(X)$ on X measures spatial correlations in cellular automaton configurations. $s^{(x)}_{(\mu)}(X)$ therefore tends to a constant if X is larger than the range of any correlations between site values. In the presence of correlations, $s^{(x)}_{(\mu)}(X)$ always decreases with X. Available data from simulations provide reliable accurate estimates for $s^{(x)}_{(\mu)}(X)$ only for $0 \leq X \lesssim 8$. Fig. 11 shows the behaviour of the equilibrium value of $s^{(x)}_{\mu}(X)$ as a function of X over this range for several typical class 3 cellular automata. For rules which yield irregular patterns the equilibrium value of $s^{(x)}_{\mu}(X)$ typically remains $\gtrsim 0.9$ for $X \lesssim 8$. $s^{(x)}_{\mu}(X)$ at equilibrium typically decreases much more rapidly for class 3 cellular automata which generate more regular patterns. At least for small X, $s^{(x)}_{\mu}(X)$ for such cellular automata typically decreases roughly as $X^{-\eta}$ with $\eta \approx 0.1$.

The values of the spatial set entropy $s^{(x)}(X)$ provide upper bounds on the spatial measure entropy $s^{(x)}_{\mu}(X)$. The distribution of nonzero probabilities $p^{(x)}_i$ for possible length X blocks is typically quite broad, yielding an $s^{(x)}_{\mu}(X)$ significantly smaller than $s^{(x)}(X)$. Nevertheless, the general behaviour of $s^{(x)}_{\mu}(X)$ with X usually roughly follows $s^{(x)}(X)$, but with a slight X delay.

As discussed in section 4, the set entropy $s^{(x)}(X)$ attains its maximum value of 1 if and only if all k^X sequences of length X appear (with nonzero probability) in evolution from some initial state. Notice that if $s^{(x)}(X) = 1$ after one time step, then $s^{(x)}(X) = 1$ at any time. In general, $s^{(x)}(X)$ takes on value 1 for blocks up to some critical length X_c (perhaps infinite), as defined in eq. (4.13).

Since a block of length X is completely determined by a sequence of length $X + 2r$ in the previous configuration, any predecessors for the block may in principle be found by an exhaustive search of all k^{X+2r} possible length $X + 2r$ sequences. The procedure for progressive construction of predecessors outlined in section 6 provides a more efficient procedure [21]. The critical block length X_c is determined by the minimum number of nodes in the finite automaton state transition graph visited on any path from the "start" to "stop" node. The state transition graph is determined by the set of transition rules $\Psi_i \rightarrow \Psi_{i+1}$. Starting with length 1 blocks, these transition rules may be found by considering construction of all possible progressively longer blocks, but ignoring blocks associated with values Ψ_i for which the transition rules have already been found. If X_c is finite, the "stop" node $\Psi = 0$ is reached in the construction of length X_c blocks. Alternatively, the state transition graph may be found to consist of closed cycles, not including $\Psi = 0$. In this case, X_c is determined to be infinite. Since the state transition graph contains at most $2^{k^{2r}}$ nodes, the value of X_c may be found after at most this many tests. The procedure thus provides a finite algorithm for determining whether all possible arbitrarily long sequences of site values may be generated by evolution with a particular cellular automaton rule.

Table II gives the critical block lengths X_c for the cellular automata illustrated in fig. 1. Class 3 cellular automata with smaller X_c tend to generate more regular patterns. Those with larger X_c presumably give systematically larger entropies and their evolution is correspondingly less irreversible.

For additive cellular automata (such as code 42 in fig. 1 and table II), all possible blocks of any length X may be reached, and have exactly k^{2r} predecessors of length $X + 2r$. In this case, therefore, evolution from a disordered initial state gives $s^{(x)}(X) = 1$ for all X (hence $X_c = \infty$). The equality of the number of predecessors for each block implies in addition in this case that $s^{(x)}_{\mu}(X) = 1$, at least for evolution from disordered initial states. Hence for additive cellular automata

$$d^{(x)} = d^{(x)}_{\mu} = 1\,. \tag{7.1}$$

Table II
Values of critical block length X_c for legal totalistic $k = 2$, $r = 2$ cellular automata as illustrated in fig. 1. For $X < X_c$, all k^X possible blocks of X site values appear with nonzero probability in configurations generated after any number of time steps in evolution from disordered initial states, while for $X \geq X_c$, some blocks are absent, so that the spatial set entropy $s^{(x)}(X) < 1$

Code	X_c	Code	X_c
2	5	32	3
4	12	34	5
6	7	36	12
8	12	38	7
10	36	40	12
12	5	42	∞
14	5	44	5
16	5	46	5
18	5	48	5
20	36	50	5
22	12	52	22
24	7	54	12
26	12	56	7
28	5	58	12
30	3	60	5

The configurations generated by additive cellular automata are thus maximally chaotic.

In general cellular automata evolving according to eq. (2.1) yield $s^{(x)}(X) = 1$ for all X, so that $d^{(x)} = 1$, if **F** is an injective (one-to-one) function of either its first or last argument (or can be obtained by composition of functions with such a property). This may be proved by induction. Assume that all the blocks of length X are reachable, with predecessors of lengths $X + 2r$. Then form a block of length $X + 1$ by adding a site at one end. To obtain all possible length $X + 1$ blocks, the value a' of this additional site must range over k possibilities. Any predecessors for length $X + 1$ blocks must be obtained by adding a $(X + 2r + 1)$-th site (with value a) at one end. For all length $X + 1$ blocks to be reachable, all values of a' must be generated when a runs over its k possible values, and the result follows. Notice that not all length $X + 1$ blocks need have the same (nonzero) number of predecessors, so that the measure entropy $s_{\mu}^{(x)}(X)$ may be less than the set entropy $s^{(x)}(X)$.

While injectivity of the rule function **F** for a cellular automaton in its first or last arguments is sufficient to give $d^{(x)} = 1$, it is apparently not necessary. A necessary condition is not known.

In section 6 it was shown that the set of configurations obtained by cellular automaton evolution for a finite number of time steps from any initial state could be specified by a regular grammar. In general the complexity of the grammar may increase rapidly with the number of time steps, potentially leading at infinite time to a set not specifiable by a regular grammar. Such behaviour may generically be expected in class 3 cellular, for which the average minimum propagation speed $\bar{\lambda} > 0$.

As discussed in section 4, one may consider the statistics of temporal as well as spatial sequences of site values. The temporal aperiodicity of the patterns generated by evolution of class 3 cellular automata from almost all initial states suggests that these systems should have nonvanishing temporal entropies $s_{(\mu)}^{(t)}(T)$ and nonvanishing temporal dimensions $d_{(\mu)}^{(t)}$. Once again, the temporal entropies for blocks starting at progressively later times quickly relax to equilibrium values. Notice that the dimension $d_{(\mu)}^{(t)}$ obtained from the large T limit of the $s_{(\mu)}^{(t)}(T)$ is always independent of the starting times for the blocks. This is to be contrasted with the spatial dimensions $d_{(\mu)}^{(x)}$, which depend on the time at which they are evaluated. Just as for spatial entropies, it found that the equilibrium temporal entropies are essentially independent of probability measure for initial configurations.

The temporal entropies $s_{(\mu)}^{(t)}(T)$ decrease slowly with T. In fact, it appears that in all cases

$$s_{(\mu)}^{(t)}(Z) \geq s_{(\mu)}^{(x)}(Z). \tag{7.2}$$

The ratio $s_{(\mu)}^{(t)}(Z)/s_{(\mu)}^{(x)}(Z)$ is, however, typically much smaller than its maximum value (4.38) equal to the maximum propagation speed λ_+. Notice that the value of λ_+ determines the slopes of the edges of triangular clearings in the patterns generated by cellular automaton evolution.

At least for the class 3 cellular automata in fig. 1 which generate irregular patterns, the equi-

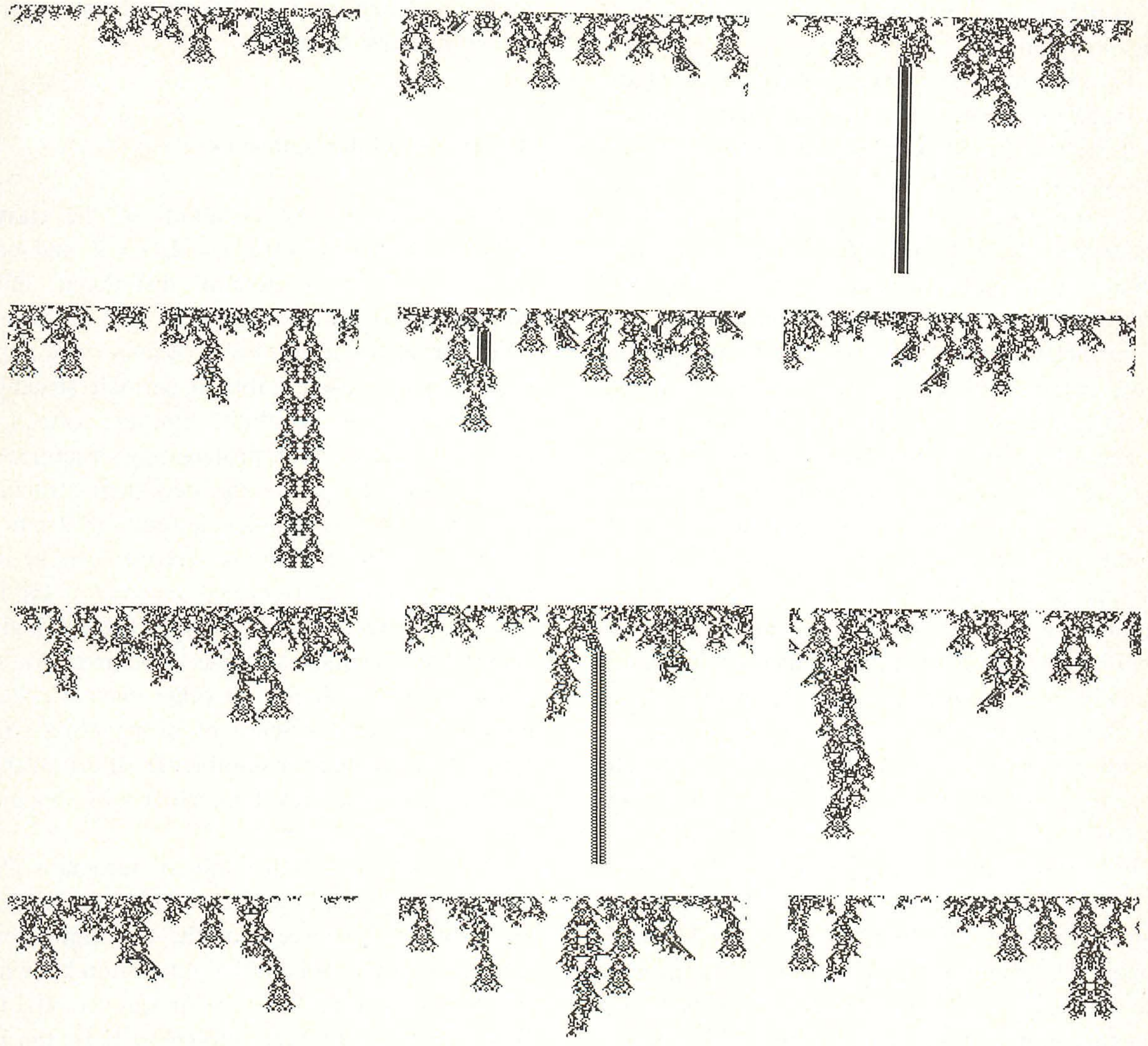


Fig. 12. Examples of the evolution of a class 4 cellular automaton (totalistic code 20 $k = 2$, $r = 2$ rule) from several disordered initial states. Persistent structures are seen to be generated in a few cases. The evolution is truncated after 120 time steps.

librium set entropy $s^{(t)}(T) = 1$ for all $T \lesssim 8$ for which data are available. Note that the result $s^{(t)}(T) = 1$ holds for all T for any additive cellular automaton rule. One may speculate that class 3 cellular automata which generate apparently irregular patterns form a special subclass, characterized by temporal dimension $d^{(t)} = 1$.

For class 3 cellular automata which generate more regular patterns, $s_{(t)}(T)$ appears to decrease, albeit slowly, with T. Just as for spatial sequences, one may consider whether the temporal sequences which appear form a set described by a regular grammar. For the particular case of the $k = 2$, $r = 1$ cellular automaton with rule number 18, there is some evidence [21] that all possible temporal sequences which contain no 11 subsequences may appear, so that $N^{(t)}(T) = F_T$ where F_T is the Tth Fibonacci number ($F_T = F_{T-1} + F_{T-2}$, $F_0 = F_1 = 1$). This implies that $N_{(t)}(T) \sim \phi^T$ ($\phi = (\sqrt{5} + 1)/2 \simeq 1.618$) for large T, suggesting a temporal set dimension $d^{(t)} = \log_2 \phi \approx 0.694$. In general, however, the set of possible temporal sequences is not expected to be described by a regular grammar.

The nonvanishing value of the average minimum propagation speed $\bar{\lambda}_-$ for class 3 cellular automata, suggests that in all cases the value of a particular site depends on an ever-increasing number of initial site values. However, the complexity of the dependence is not known. The value of a site after t time steps can always be specified by a table with an entry for each of $k^{2\lambda+t}$ relevant initial sequences. Nevertheless, it is possible that a finite state automaton, specified by a finite state transition graph, could determine the value of sites at any time

The behaviour of finite class 3 cellular automata with additive rules was analysed in some detail in ref. 2. It was shown there that the maximal cycle length for additive cellular automata grows on average exponentially with the size N of the cellular automaton. Most cycles were found to have maximal length, and the number of distinct cycles was found also to grow on average exponentially with N. The lengths of transients leading to cycles was found to grow at most linearly with N. The fraction of states on cycles was found on average to tend a finite limit.

For most class 3 cellular automata, the average cycle length grows quite slowly with N, although in some cases, the absolute maximum cycle length appears to grow rapidly. The lengths of transients are typically short for cellular automata which generate more regular patterns, but often become very long as N increases for cellular automata which generate more irregular patterns. The fractions of states on cycles are typically much larger for finite class 3 cellular automata which generate irregular patterns than for those which generate more regular patterns. This is presumably a reflection of the lower irreversibility and larger attractor dimension found for the former case in the infinite size limit.

8. Class 4 cellular automata

Fig. 12 shows the evolution of the class 4 cellular automaton with $k=2$, $r=2$ and code number 20, from several disordered initial configurations. In most cases, all sites are seen to "die" (attain value zero) after a finite time. However, in a few cases, stable or periodic structures which persist for an infinite time are formed. In addition, in some cases, propagating structures are formed. Fig. 13 shows the persistent structures generated by this cellular automaton from all initial configurations whose nonzero sites lie in a region of length 20 (reflected versions of the last three structures are also found). Table III gives some characteristics of these structures. An important feature, shared by other class 4 cellular automata, is the presence of propagating structures. By arranging for suitable reflections of these propagating structures, final states with any cycle lengths may be obtained.

The behaviour of the cellular automata illustrated in fig. 13, and the structures shown in fig. 14 are strongly reminiscent of the two-dimensional (essentially totalistic) cellular automaton known as the "Game of Life"* (for references see [1]). The Game of Life has been shown to have the important property of computational universality. Cellular automata may be viewed as computers, in which data represented by initial configurations is processed by time evolution. Computational universality (e.g. [15–18]) implies that suitable initi configurations can specify arbitrary algorithm procedures. The system can thus serve as a gener purpose computer, capable of evaluating a (computable) function. Given a suitable encoding, the system may therefore in principle simulate any other system, and in this sense may be considered capable of arbitrarily complicated behaviour.

The proof of computational universality for the Game of Life [22] uses the existence of cellular

*Each site in this cellular automaton can take on one of two possible values; the time evolution rule involves nine site (type II) neighbourhoods. If the values of less than 2 or more than 3 of the eight neighbours of a particular site are nonzero then the site takes on value 0 at the next time step; if 2 neighbouring sites are nonzero the site takes the same value as on the previous time steps; if exactly 3 neighbouring sites are nonzero, the site takes on value 1.

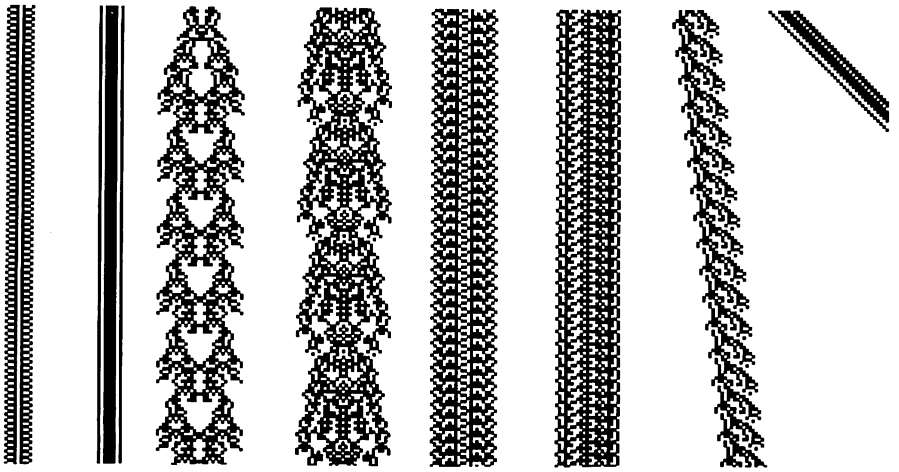


Fig. 13. Persistent structures found in the evolution of the class 4 cellular automaton illustrated in fig. 12 from initial states with nonzero sites in a region of 20 or less sites. Reflected versions of the last three structures are also found. Some properties of the structures are given in table III. These structures are almost sufficient to provide components necessary to demonstrate a universal computation capability for this cellular automaton.

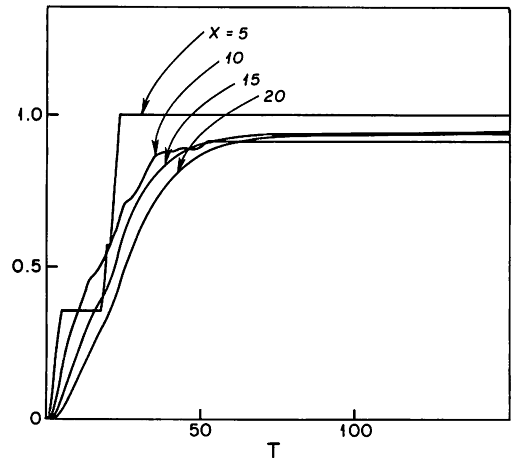


Fig. 14. Fraction of configurations in the class 4 cellular automaton of figs. 12 and 13 which evolve to the null configuration after T time steps, from initial states with nonzero sites in a region of length less than X (translates of configurations are not included). The asymptotic "halting probability" is around 0.93; 7% of initial configurations generate the persistent structures of fig. 13 and never evolve to the null configuration.

automaton structures which emulate components (such as "wires" and "NAND gates") of a standard digital computer. The structures shown in fig. 14 represent a significant fraction of those necessary. A major missing element is a configuration (dubbed the "glider gun" in the Game of Life) which acts like a clock, and generates an infinite sequence of propagating structures. Such a configuration would involve a finite number of initial nonzero sites, but would lead to unbounded growth, and an asymptotically infinite number of nonzero sites. There are however indications that the required initial configuration is quite large, and is very difficult to find.

These analogies lead to the speculation that class 4 cellular automata are characterized by the capability for universal computation. $k = 2$, $r = 1$ cellular automata are too simple to support universal computation; the existence of class 4 cellular automata with $k = 2$, $r = 2$ (cf. figs. 12 and 13) and $k = 3$, $r = 1$ suggests that with suitable time evolution rules even such apparently simple systems may be capable of universal computation.

There are important limitations on predictions which may be made for the behaviour of systems capable of universal computation. The behaviour of such systems may in general be determined in detail essentially only by explicit simulation of their time evolution. It may in general be predicted using other systems only by procedures ultimately equivalent to explicit simulation. No finite algo-

Table III
Persistent structures arising from initial configurations with length less than 20 sites in the class 4 totalistic cellular automaton with $k = 2$, $r = 2$ and code number 20, illustrated in figs. 12, 13 and 14. $\phi(X)$ gives the fraction of initial configurations with nonzero sites in a region less than X sites in length which generate a particular structure. When an initial configuration yields multiple structures, each is included in this fraction.

Period	Minimal predecessor	$\phi(10)$	$\phi(20)$
2	10010111 (151)	0.027	0.024
9R	10111011 (187)	0.012	0.0061
1	10111101 (189)	0.014	0.0075
22	11000011 (195)	0.018	0.017
9L	11011101 (221)	0.012	0.0061
1R	1001111011 (635)	0.0020	0.00066
1L	1101111001 (889)	0.0020	0.00066
38	11110100100101111 (125231)	0	2.9×10^{-5}
4	10010001011011110111 (595703)	0	7.6×10^{-6}
4	10010101001010110111 (610999)	0	7.6×10^{-6}
4	10011000011111101111 (624623)	0	7.6×10^{-6}

rithm or procedure may be devised capable of predicting detailed behaviour in a computationally universal system. Hence, for example, no general finite algorithm can predict whether a particular initial configuration in a computationally universal cellular automaton will evolve to the null configuration after a finite time, or will generate persistent structures, so that sites with nonzero values will exist at arbitrarily large times. (This is analogous to the insolubility of the halting problem for universal Turing machines (e.g. [15–18]).) Thus if the cellular automaton of figs. 12 and 13 is indeed computationally universal, no finite algorithm could predict whether a particular initial state would ultimately "die", or whether it would ultimately give rise to one of the persistent structures of fig. 13. The result could not be determined by explicit simulation, since an arbitrarily large time might elapse before one of the required states was reached. Another universal computer could also in general determine the result effectively only by simulation, with the same obstruction.

If class 4 cellular automata are indeed capable of universal computation, then their evolution involves an element of unpredictability presumably not present in other classes of cellular automata. Not only does the value of a particular site after many time steps potentially depend on the values of an increasing number of initial site values; in addition, the value cannot in general be determined by any "short-cut" procedure much simpler than explicit simulation of the evolution. The behaviour of a class 4 cellular automaton is thus essentially unpredictable, even given complete initial information: the behaviour of the system may essentially be found only by explicitly running it.

Only infinite cellular automata may be capable of universal computation; finite cellular automata involve only a finite number of internal states, and may therefore evaluate only a subset of all computable functions (the "space-bounded" ones).

The computational universality of a system implies that certain classes of general predictions for its behaviour cannot be made with finite algorithms. Specific predictions may nevertheless often be made, just as specific cases of generally non-computable function may often be evaluated. Hence, for example, the behaviour of all configurations with nonzero sites in a region of length 20 or less evolving according to the cellular automaton rules illustrated in figs. 12 and 13 has been completely determined. Fig. 14 shows the fraction of initial configurations which evolve to the null state within T time steps, as a function of T, for various sizes X of the region of nonzero sites. For large X and large T, it appears that the fraction of configurations which generate no persistent structures (essentially the "halting probability") is approximately 0.93. It is noteworthy that the curves in fig. 14 as a function of T appear to approach a fixed form at large X. One may speculate that some aspects of the form of such curves may be universal to all systems capable of universal computation.

The sets of persistent structures generated by class 4 cellular automata typically exhibit no simple patterns, and do not appear to be specified, for example, by regular grammars. Specification of persistent structures by a finite procedure is necessarily impossible if class 4 cellular automata are indeed capable of universal computation. Strong support of the conjecture that class 4 cellular automata are capable of universal computation would be provided by a demonstration of the equivalence of systematic enumeration of all persistent structures in particular class 4 cellular automata to the systematic enumeration of solutions to generally insoluble Diophantine equations or word problems.

Although one may determine by explicit construction that specific cellular automata are capable of universal computation, it is impossible to determine in general whether a particular cellular automaton is capable of universal computation. This is a consequence of the fact that the structures necessary to implement universal computation may be arbitrarily complicated. Thus, for example, the smallest propagating structure might involve an arbitrarily long sequence of site values.

For class 1, 2 and 3 cellular automata, fluctuations in statistical quantities are typically found to become progressively smaller as larger numbers of sites are considered. Such systems therefore exhibit definite properties in the "infinite volume" limit. For class 4 cellular automata, it seems likely that fluctuations do not decrease as larger number of sites are considered, and no simple smooth infinite volume limit exists. Important qualitative effects can arise from special sequences appearing with arbitrarily low probabilities in the initial state. Consider for example the class 4 cellular automaton illustrated in figs. 12 and 13. The evolution of the finite sequences in this cellular automaton shown in fig. 12 (and many thousands of other finite sequences tested) suggests that the average density of nonzero sites in configurations of this cellular automaton should tend to a constant at large times. However, in a sufficiently long finite initial sequence, there should exist a subsequence from which a "glider gun" structure evolves. This structure would generate an increasing number of nonzero sites at large times, and its presence would completely change the average large time density. As a more extreme example, it seems likely that a sufficiently long (but finite) initial sequence should evolve to behave as a self-reproducing "organism", capable of eventually taking over its environment, and leading to completely different large time behaviour. Very special, and highly improbable, initial sequences may thus presumably result in large changes in large time properties for class 4 cellular automata. These sequences must appear in a truly infinite (typical) initial configuration. Although their density is perhaps arbitrarily low, the sequences may evolve to structures which come to dominate the statistical properties of the system. The possibility of such phenomena suggest that no smooth infinite volume exists for class 4 cellular automata.

Some statistical results may be obtained from large finite class 4 cellular automata, although the results are expected to be irrelevant in the truly infinite volume limit. The evolution of most class 4 cellular automata appears to be highly irreversible*. This irreversibility is reflected in the small set of persistent structures usually generated as end-products of the evolution. Changes in small regions of the initial state may affect many sites at

*This feature allows practical simulation of such cellular automata to be made more efficient by storing information on the evolution of the specific sequences of sites which occur with larger probabilities (cf. [23]).

large times. There are however very large fluctuations in the propagation speed, and no meaningful averages may be obtained. It should be noted that groups of class 4 cellular automata with different rules often yield qualitatively similar behaviour, and similar sets of persistent structures, suggesting further classification.

The frequency with which a particular structure is generated after an infinite time by the evolution of a universal computer from random (disordered) input gives the "algorithmic probability" p_A [24] for that structure. This algorithmic probability has been shown to be invariant (up to constant multiplicative factors) for a wide class of universal computers. In general, one may define an "evolutionary probability" $p_E(t)$ which gives the probability for a structure to evolve after t time steps from a random initial state. Complex structures formed by cellular automata will typically have evolutionary probabilities which are initially small, but later grow. As a simple example, the probability for the sequence which yields a period 9 propagating structure in the cellular automaton of figs. 12 and 13 begins small, but later increases to a sufficiently large value that such structures are almost always generated from disordered states of 2000 or more sites. In a much more complicated example, one may imagine that the probability for a self-reproducing structure begins small, but later increases to a substantial value. Structures whose evolutionary probability becomes significant only after a time $>T$ may be considered to have "logical depth" [25] T.

9. Discussion

Cellular automata are simple in construction, but are capable of very complex behaviour. This paper has suggested that a considerable universality exists in this complex behaviour. Evidence has been presented that all one-dimensional cellular automata fall into four basic classes. In the first class, evolution from almost all initial states leads ultimately to a unique homogeneous state. The second class evolves to simple separated structures. Evolution of the third class of cellular automata leads to chaotic patterns, with varying degrees of structure. The behaviours of these three classes of cellular automata are analogous to the limit points, limit cycles and chaotic ("strange") attractors found in continuous dynamical systems. The fourth class of cellular automata exhibits still more complicated behaviour, and its members are conjectured to be capable of universal computation.

Even starting from disordered or random initial configurations, cellular automata evolve to generate characteristic patterns. Such self-organizing behaviour occurs by virtue of the irreversibility of cellular automaton evolution. Starting from almost any initial state, the evolution leads to attractors containing a small subset of all possible states. At least for the first three classes of cellular automata, the states in these attractors form a Cantor set, with characteristic fractal and other dimensions. For the first and second classes, the states in the attractor may be specified as sentences with a regular grammar. For the fourth class, the attractors may be arbitrarily complicated, and no simple statistical characterizations appear possible.

The four classes of cellular automata may be distinguished by the level of predictability of their "final" large time behaviour given their initial state. For the first class, all initial states yield the same final state, and complete prediction is trivial. In the second class, each region of the final state depends only on a finite region of the initial state; knowledge of a small region in the initial state thus suffices to predict the form of a region in the final state. In the evolution of the third class of cellular automata, the effects of changes in the initial state almost always propagate forever at a finite speed. A particular region thus depends on a region of the initial state of ever-increasing size. Hence any prediction of the "final" state requires complete knowledge of the initial state. Finally, in the fourth class of cellular automata, regions of the final state again depend on arbitrarily large regions of the initial state. However, if cellular automata in the class are indeed capable of universal computation, then this dependence may be arbitrarily complex, and the behaviour of the system can be found by no procedure significantly simpler than direct simulation. No meaningful prediction is therefore possible for such systems.

Acknowledgements

I am grateful to many people for discussions, including C. Bennett, J. Crutchfield, D. Friedan, P. Gacz, E. Jen, D. Lind, O. Martin, A. Odlyzko, N. Packard, S^2. Shenker, W. Thurston, T. Toffoli and S. Willson. I am particularly grateful to J. Milnor for extensive discussions and suggestions.

References

[1] S. Wolfram, 'Statistical mechanics of cellular automata", Rev. Mod. Phys. 55 (1983) 601.

[2] O. Martin, A.M. Odlyzko and S. Wolfram, "Algebraic properties of cellular automata", Bell Laboratories report (January 1983); Comm. Math. Phys., to be published.

[3] D. Lind, "Applications of ergodic theory and sofic systems to cellular automata", University of Washington preprint (April 1983); Physica 10D (1984) 36 (these proceedings).

[4] S. Wolfram, "CA: an interactive cellular automaton simulator for the Sun Workstation and VAX", presented and demonstrated at the Interdisciplinary Workshop on Cellular Automata, Los Alamos (March 1983).

[5] T. Toffoli, N. Margolus and G. Vishniac, private demonstrations.

[6] P. Billingsley, Ergodic Theory and Information (Wiley, New York, 1965).

[7] D. Knuth, Seminumerical Algorithms, 2nd. ed. (Addison-Wesley, New York, 1981), section 3.5.

[8] R.G. Gallager, Information Theory and Reliable Communications (Wiley, New York, 1968).

[9] J.D. Farmer, "Dimension, fractal measures and the probabilistic structure of chaos", in: Evolution of Order and Chaos in Physics, Chemistry and Biology, H. Haken, ed. (Springer, Berlin, 1982).

[10] J.D. Farmer, private communication.

[11] B. Mandelbrot, The Fractal Geometry of nature (Freeman, San Francisco, 1982).

[12] J.D. Farmer, "Information dimension and the probabilistic structure of chaos", Z. Naturforsch. 37a (1982) 1304.

[13] P. Grassberger, to be published.

[14] P. Diaconis, private communication; C. Stein, unpublished notes.

[15] F.S. Beckman, "Mathematical Foundations of Programming (Addison-Wesley, New York, 1980).

[16] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation (Addison-Wesley, New York, 1979).

[17] Z. Manna, Mathematical Theory of Computation (McGraw-Hill, New York, 1974).

[18] M. Minsky, Computation: Finite and Infinite Machines (Prentice-Hall, London, 1967).

[19] B. Weiss, "Subshifts of finite type and sofic systems", Monat. Math. 17 (1973) 462. E.M. Coven and M.E. Paul, "Sofic systems", Israel J. Math. 20 (1975) 165.

[20] P. Grassberger, "A new mechanism for deterministic diffusion", Wuppertal preprint WU B 82–18 (1982).

[21] J. Milnor, unpublished notes.

[22] R.W. Gosper, unpublished; R. Wainwright, "Life is universal!", Proc. Winter Simul. Conf., Washington D.C., ACM (1974). E.R. Berlekamp, J.H. Conway and R.K. Guy, Winning Ways, for Your Mathematical Plays, vol. 2 (Academic Press, New York, 1982), chap. 25.

[23] R.W. Gosper, "Exploiting regularities in large cellular spaces", Physica 10D (1984) 75 (these proceedings).

[24] G. Chaitin, "Algorithmic information theory", IBM J. Res. & Dev., 21 (1977) 350; "Toward a mathematical theory of life", in: The Maximum Entropy Formalism, R.D. Levine and M. Tribus, ed. (MIT press, Cambridge, MA, 1979).

[25] C. Bennett, "On the logical "depth" of sequences and their reducibilities to random sequences", IBM report (April 1982) (to be published in Info. & Control).

Wolfram

keep all numeric keys

Buy numerical keypad

+ 10 function keys on lhs
+ numeric pad on rhs.

Number + enter key needed.

Numeric keypad;

slides mostly
640 x 480 Slides 320 x 240 lines

Points

1. Some are Self-similar, eg Top right corner pg 199
2. Simple initial state + simple rule → complex pattern.
eg pg 199 2nd row
3. 2 initial states of each.

609 683 0436

~~Margolus'~~ Cell Vichniac for 2D CA's:

Color pics out of proceeding 1983 conf.
6 pictures on a page:
✓ 1. Version of 'life' fluid flow.
✓ 2. Waves sending out natural waves
✓ 3. Snowflake;

Natural systems in story in 2D case; → natural systems,

Cellular Automata

Wolfram Sept 11 '84

Uses: 1. Natural phenomena   eg Snowflake (2D)
Seashells
Bacteria
1D Shock waves

2. Experimental Maths

1. Self-similar; in principle could have done analytically, but was clearly shown experimentally

2. General classification - 4 classes
discovered by looking at pictures
influence of small change

Simplest known examples to construct that give complex behaviour

→ more complicated with time. cf 2nd law → max entropy

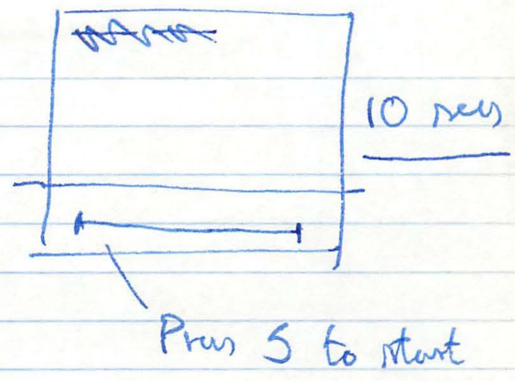
Bridge between computing & diff equations

Do image processing - local rules

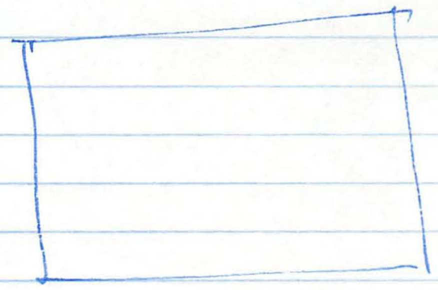
1. Idle mode

Fixed resolution
124 resolution

Draw CA; 1D + 2D
Random initial states + random
Invite interaction initial states

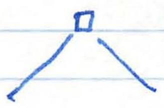


2. Intro to cellular automata



Arrowhead curve

Rule in English



Mod 2 Rule

a. Try this rule with another initial state:

1. ◻ ◻

2. ◻◻ ◻◻

3 ◻◻ ◻◻ ◻◻ ◻

Choose 1, 2 or 3.

4. Random: ◻ ◻ ◻ ◻◻ ◻◻ ◻ ◻◻

~~3~~.6. Rules

1/2

You've seen one rule (repeat it):

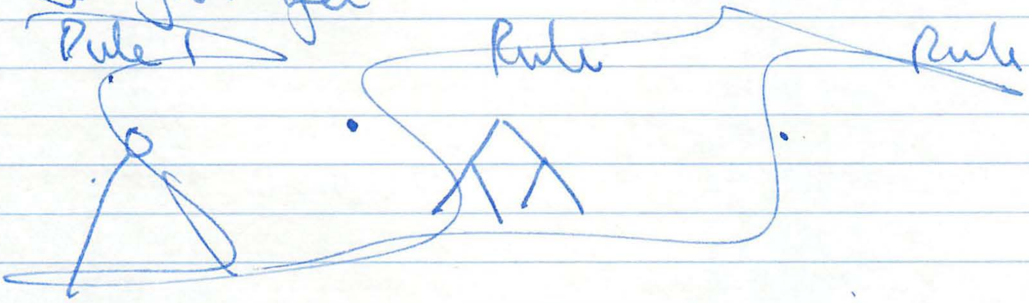
Many rules.

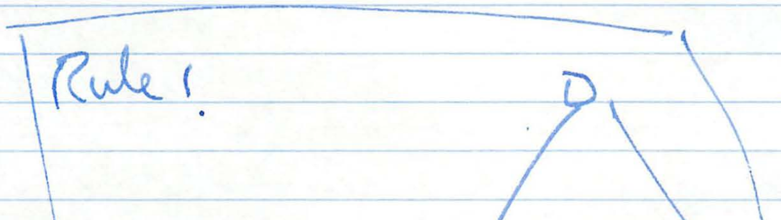
~~Here~~ This program shows rules which depend on the sum of the cell itself and its ~~to~~ neighbours to the left and right.

The rules affect the pattern radically.

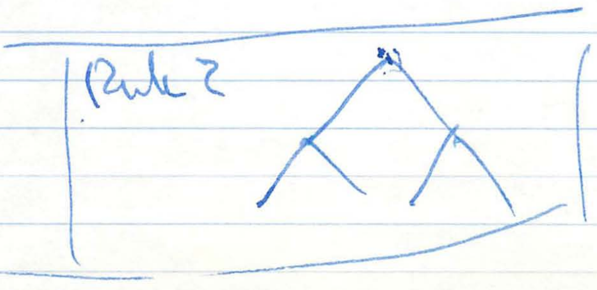
For example; with the initial configuration of one cell 'on':

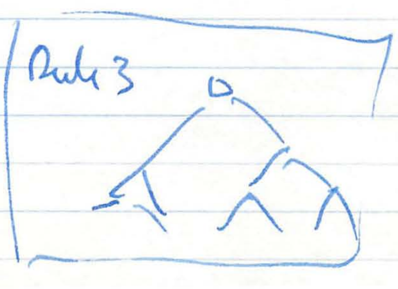
~~I~~ you get





Pause ~~5~~ 2 secs





~~Note if dies out quickly, skip rule~~

NS) ~~Do you want to see some random rules applied to this starting? or ↓ specify your own?~~

Do you want to choose a rule ~~now~~ or let the program choose ~~a ru~~ one?

1. You choose
2. ~~Program~~ Computer chooses → execute immediately

Note ½ randomly chosen
½ preselected goodies
Carry on when press return or time out after X mins

1. ~~There are 64, . . . rules.~~

This ~~program~~ computer is set up to show rules of a particular kind (~~B rules~~ they depend . . .)

There are 6 . . . . of them.

Enter a number between 0 and —
~~or~~ and press return; ~~to choose~~.

~~8~~.

after the 3rd ~~time around~~ rule reduction :
or after 5 ~~10~~ times of computer choosing it :
NS > ~~A~~ ~~Noo~~ [illegible]

You can now choose initial config's also.

1. Random ◻ o . . . . . .

2. oooo

3. o.

From this on, always offer both.

<u>Note all Game 4.</u>

Physica 10D (1984) 81–95
North-Holland, Amsterdam

PHYSICS-LIKE MODELS OF COMPUTATION*

Norman MARGOLUS
MIT Laboratory for Computer Science, Cambridge Massachusetts 02139, USA

Reversible Cellular Automata are computer-models that embody discrete analogues of the classical-physics notions of space, time, locality, and microscopic reversibility. They are offered as a step towards models of computation that are closer to fundamental physics.

1. Introduction

Reversible Cellular Automata (RCA) are computer-models that embody discrete analogues of the classical-physics notions of space, time, locality, and microscopic reversibility.

In this paper, I will describe some RCA, explain how they can be used as computer models, and discuss RCA analogues of energy and entropy – concepts that are fundamental in physics, but have not played a fundamental role in computer theory.

2. Cellular automata

In CA, 'space' is a regular lattice of 'cells', each of which contains one of a small allowed set of integers. Only cells that are close together interact in one 'time-step' – the time evolution is given by a rule that looks at the contents of a few neighbouring cells, and decides what should change. At each step, this local rule is applied everywhere simultaneously[10].

The best-known example of such a 'digital-world' is Conway's[5] "Game of Life". On a sheet of graph-paper, fill each cell with a '1' or a '0'. In each three-by-three neighbourhood there is a center cell and eight adjacent cells. The new state of each cell is determined by counting the number of adjacent 1's – if exactly two adjacent cells contain a one, the center is left unchanged. If three are ones, the center becomes a one. In all other cases, the center becomes a zero.

Such a rule gives rise to a set of characteristic patterns that 'move' (reappear in a slightly displaced position after some number of steps) patterns that are stable (unchanging with time) patterns that oscillate (pass through some cycle of configurations) and many very complicated interactions and behaviours. The evolution of a given initial configuration is often very hard to anticipate (see colour plate in [9]).

One way to show that a given rule can exhibit complicated behaviour is to show (as has been done for "Life"[4]) that in the corresponding 'world' it is possible to have computers. If you start the automaton with an appropriate initial state, you will see digits acting as signals moving about and interacting with each other to perform all of the logical operations of a digital computer. Such a computer-automaton is said to be *universal*.**

* This research was supported in part by the Defense Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contracts Nos. N00014–75-C-0661 and N00014–83–K–0125, and in part by NSF Grant No. 8214312-IST.

** Von Neumann[10] was interested in the problem of evolution – could life emerge from simple rules? He exhibited a CA rule that permitted computers, and in which these computers could reproduce and mutate. In this paper, I refer only to the existence of computers when I use the term universal.

3. Reversible cellular automata

Any CA rule can be described by an equation of the form*

$$S_{i,t+1} = f(S_{\{i\},t}), \qquad (1)$$

where $S_{i,t+1}$ is the state of the cell at position 'i' and at time '$t+1$', and $f(S_{\{i\},t})$ is a function of the states of cells in a neighbourhood of i, at time t.

In general, (1) gives rise to a non-invertible dynamics. If f is the 'Life' rule, this evolution is not reversible – if an area now contains only zeros, did it contain zeros one step ago, or were there perhaps some isolated ones that just changed? Its impossible to tell.

It turns out to be very easy to write down CA laws that give an invertible dynamics – just as easy as constructing irreversible ones, in fact. Consider first the following finite difference equation, with x_t a real variable:

$$x_{t+1} = f(x_t) - x_{t-1}. \qquad (2)$$

If you want to compute x_{t+1}, you must know x_t and x_{t-1} – these two constitute the complete 'state' of the system. For what functions f will the time evolution be invertible?

$$x_{t-1} = f(x_t) - x_{t+1}, \qquad (3)$$

therefore any f at all will do**! Knowing x for two consecutive times allows you to calculate any preceding or any succeeding value of x (To my knowledge Fredkin [2] was the first to study reversibility in finite-difference-equations of this sort.)

The generalization to CA is straightforward – let x in (2) be replaced by c_i, the contents of the cell at position 'i' in our automaton,

$$c_{i,t+1} = f(c_{\{i\},t}) - c_{i,t-1}, \qquad (4)$$

where $f(c_{\{i\},t})$ is any function involving the contents of cells near position 'i', at time 't', and the difference is taken mod the number of allowed cell values***. If we let the state of a cell correspond to its contents in two successive steps, then (4) can be reexpressed in the form (1), but its reversibility is not manifest†.

Such rules can be universal (I give an example in the appendix). Reversible computation is a relatively new idea [1, 3, 8] that has been used to show that a fundamental lower bound on dissipation in computers associated with the irreversibility of conventional logic elements [6] can be avoided.

4. Entropy in RCA

If we fill the cells of our automaton with randomly chosen binary values and then evolve it according to the Life rule, we see a complex ebb and flow of structures and activity, with so-called 'gliders' arising here and there, moving across clumps of zeros, and then being drawn back into a complex boiling 'soup' of activity, or perhaps rekindling complicated interactions in an area which had settled down into uncoupled, short period oscillating structures.

If, instead of the Life rule, we follow some invertible time evolution, we invariably find that, at each step, the state of the automaton looks just as random as when we started‡. This is expected

*This serves to clarify what sorts of systems we're dealing with, but is often not the simplest or most illuminating way to express the rule.

**Assuming integer addition and subtraction is done without error, if such an equation is iterated on a digital computer, its time evolution remains *exactly* reversible, despite roundoff and truncation errors in computing f.

***Differences mod-k and logical functions can always be re-expressed as ordinary polynomial functions. For example, if A and B are binary variables, then $(A-B)^2$ is the same as $A+B(\text{mod } 2)$, $1-A$ is the same as not(A), $A*B$ is the same as and(A, B), etc. Thus (4) is equivalent to an ordinary real-variable finite difference equation with integer initial conditions.

†The global time evolution generated by (4) is not guaranteed to be invertible unless suitable boundary conditions are chosen, such as no boundary (i.e. an infinite or periodic space) or 'fixed' boundaries (cell values on the boundary are not allowed to change with time).

‡Spatial correlations will not arise if they are initially absent, but time correlations are often very evident, and are characteristic of the particular rule being employed – see the next section.

from a simple counting argument, since most configurations look random (only a very few random-looking initial configurations can be mapped by a given number of invertible steps into the few simple-looking configurations, since the overall mapping is bijective).

This is not meant to imply that RCA are less interesting than irreversible CA. Starting an RCA from a random state is like starting a thermodynamic system in a maximum entropy state – its not allowed to get any simpler since its randomness can't decrease, and it can't get more complicated, since its already as random as it can be, and so nothing much happens.

If we start an RCA from a very non-random state (e.g. some small pattern on a background of zeros) then we can have an interesting time evolution. If we choose a rule and an initial state that allow information to propagate, then what tends to happen is that the state of the RCA becomes more and more complicated. More precisely, if each state of the automaton is viewed as a 'message', with the contents of the cells being the characters of the message, and if only local measures of correlation are applied, then the amount of information* in successive messages is increasing. Of course the automaton is really only repeatedly encrypting its state, and so if all correlations are taken into account the amount of information really never changes. What happens is that the automaton will introduce some redundancy into the message, and use more cells to encode the same information. Information that was initially localized becomes spread out as correlations between the states of many cells, and it becomes very difficult for a locally invertible evolution to put the redundant pieces back together**. To use an analogy, an invertible mapping could change two copies of this document into one copy, and several sheets of blank paper. Two separate invertible mappings, each acting only on one of the copies, could not accomplish this end.

From the point of view of creatures 'living' inside an RCA, their inability to make use of complicated correlations between large numbers of cells means that for all practical purposes, the entropy of the automaton increases. To use a thermodynamic analogy, if I want to compress a gas, it doesn't help me to know that the gas was all in one corner of the room just a few minutes ago. I have no ability to make use of the complicated correlations that this statement implies, and so I say that entropy increased when the gas expanded to its current volume.

5. Conservation laws in second-order RCA

In general, an RCA has as many conserved quantities as there are cells – it 'remembers' the initial state of each cell, since you can recover this information by running the system backwards. Do these give rise to any invariants which can be computed in a local manner from the current state of the system? Can we find an invariant that is analogous to a classical mechanical energy?

In RCA, the simplest locally-computable invariants are of course cells whose values never change***. Such situations can arise because many rules ignore the remainder of the neighbours when part of the neighbourhood has some particular configuration†. For example, consider any rule that, in all cases where the center cell of the neighbourhood is 1, ignores the rest of the neighbours and returns a 2. Such a rule, when used with (4), results in a very simple conservation law. If we look at the case in one dimension where the automaton at two consecutive time-steps looks like

*For a discussion of the information content of a message, cf. [7].

**(4) generates a locally invertible time evolution. If we know the values of cells near position i at two successive times, we can tell what the preceding value of the center cell was.

***In mechanics, this corresponds to degrees of freedom that, for certain initial conditions, are decoupled from the rest.

†For rules with 2 states per cell, only two rules, "count the parity of the neighbourhood" and its complement, have no configuration of part of the neighbourhood that makes the remaining neighbours irrelevant.

this:

$$\begin{array}{lll} t-1 & \ldots 1 \ldots & \text{'.' indicates a cell whose value is} \\ t & \ldots 1 \ldots & \text{irrelevant to the discussion} \end{array} \qquad (5)$$

then the center cell here will always be a 1.

For a more interesting 1-dimensional example, consider a 2 state per cell CA with a rule f that returns a 1 iff each of the two cells adjacent to the center is the same as the center:

$$f(c_{\{x\},t}) = \begin{cases} 1, & \text{if } c_{x-1,t} = c_{x,t} = c_{x+1,t} \\ 0, & \text{otherwise.} \end{cases} \qquad (6)$$

* In irreversible CA, a guarantee that a cell will always be part of such a pair does not guarantee that it always has been.

** An extreme instance of 'decoupling' of entire regions occurs with any rule that doesn't depend on the center cell, but depends on its nearest neighbours. For example, in 1D we might have a region that looks like this:

$$\begin{array}{ll} t-1 & \ldots 1.1.0.1 \ldots \\ t & \ldots 1.0.0.1.1 \ldots \\ t+1 & \ldots ?.?.?.? \ldots \end{array}$$

From (4) it is clear that we have enough information to compute the states of the cells marked with '?' – the system decouples into two entirely independent (but interleaved) sublattices, each evolving without reference to the other.

With this rule, 'a' and 'b' standing for any binary values, and '$\bar{a}$', '$\bar{b}$' their binary complements, the second-order time evolution given by (4) says that

$$\left\{ \begin{array}{ll} t-1 & \ldots a\bar{a} \ldots \\ t & \ldots b\bar{b} \ldots \end{array} \right\} \rightarrow \left\{ \begin{array}{ll} t & \ldots b\bar{b} \ldots \\ t+1 & \ldots a\bar{a} \ldots \end{array} \right\} \qquad (7)$$

which is again of the same form, so these two cells are decoupled from the rest of the automaton. Any cell which is *not* initially part of such a pair will never be (and never was)*; counting all such cells gives us an (invariant) estimate of how many cells are available to represent dynamically changing information (but only an estimate – whole regions may be decoupled from the rest of the automaton because they are surrounded by a wall of decoupled cells**; a local counting wouldn't reveal this).

If we concentrate on the active (as opposed to the decoupled) cells, we can distinguish various kinds of activity, and try to associate conserved quantities with each. As a simple 1-dimensional example, consider 'dislocations' propagating in a regular background pattern of cells. Using the rule (6) again, consider the sequence of steps shown in fig. 1 (light and dark squares stand for 0's and 1's respectively; dislocations are triplets in a background of pairs and are outlined for emphasis). In this evolution, the number of such 'signals' is the

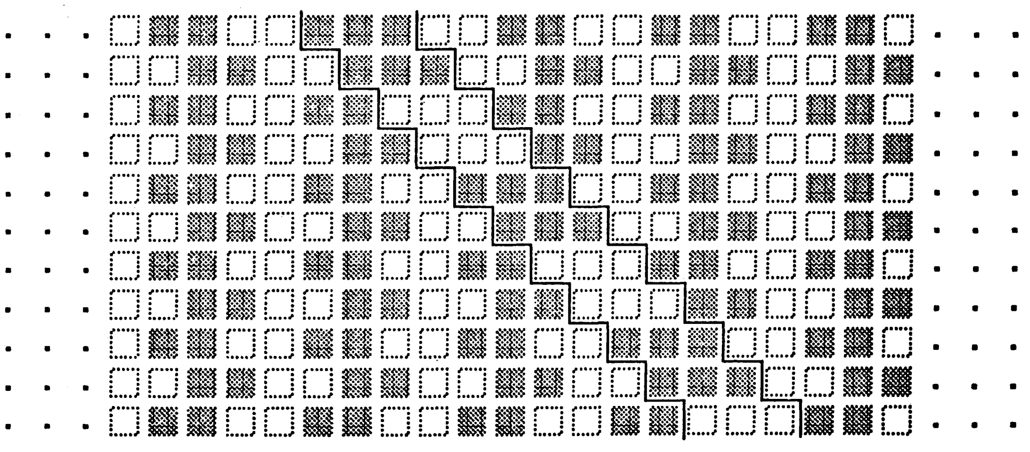


Fig. 1.

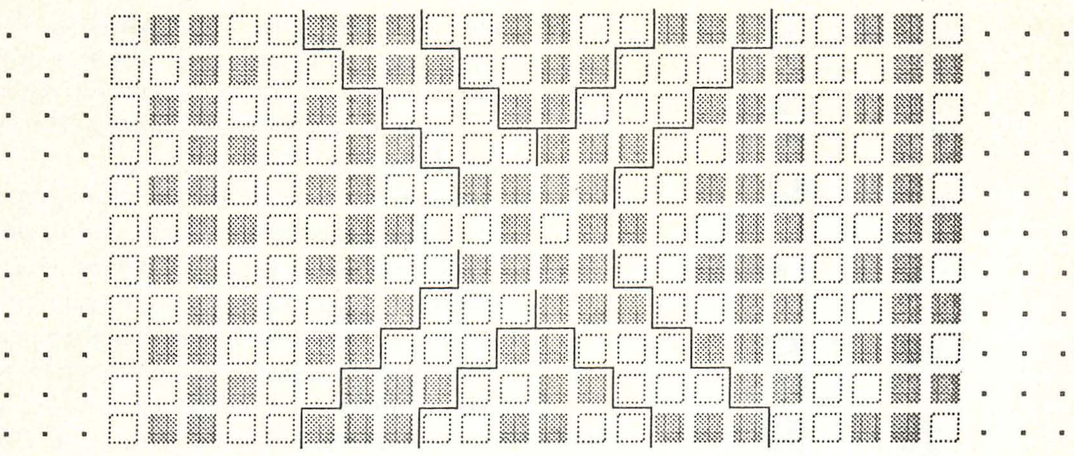


Fig. 2.

same as the number of blocks of cells we start off with the form $\begin{smallmatrix}\cdots aa\cdots\\ \cdots bb\cdots\end{smallmatrix}$, and is conserved.

If two such dislocations collide, we know that they can't just completely stop moving. Proof: if we tried to invert the evolution, we wouldn't know when to start the signals moving again. In fig. 2, the following quantity is the same for every pair of consecutive time-steps:

$$\left(\#\text{ of cells in blocks of form }\begin{matrix}\ldots aa\ldots\\ \ldots bb\ldots\end{matrix}\right) + \left(\#\text{ of cells in blocks of form }\begin{matrix}\ldots c\,.\,c\ldots\\ \ldots d\,.\,d\ldots\end{matrix}\right) \tag{8}$$

The first term counts the number of moving signals, and so could be thought of as a 'kinetic-energy' analogue. The second term accounts for the disappearance of this 'K.E.' during a collision, and so could be considered a potential-energy analogue.

6. First-order RCA

Rather than continue to analyze RCA in order to discover conservation laws, we will now proceed to construct a class of automata that all obey a very simple local conservation law: the total number of 1's never changes, and neither does the number of 0's.

The trick we will use is quite general, but it will be illustrated in 2 dimensions with 2 states per cell. Fig. 3 shows a Cartesian lattice of cells, divided into 2×2 blocks of cells. We treat each 2×2 block as a conservative-logic[3] gate, with 4 inputs (its current state) and 4 outputs (its next state). These 'gates' are interconnected in an entirely uniform and predictable manner – in applying the rule to the 2×2 blocks, we alternate between using the solid blocking in this diagram for one step, and then using the dotted blocking for the next*. Fig.

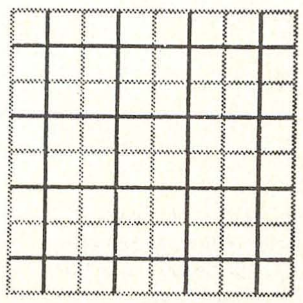


Fig. 3.

* If f_s is the global rule that applies to the solid blocking, and f_d to the dotted blocking, then $S_{t+1} = f_s(f_d(S_t))$ describes the evolution using a time independent rule. By including a small amount of positional information in the state of each cell, this rule can be written in the form(1).

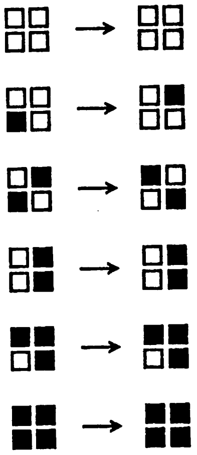


Fig. 4.

4 shows an example of a conservative rule (one that conserves 1's and 0's) that is reversible. In the case of all 0's or all 1's, there is no choice, they remain unchanged. Any rotation of one of the blocks on the left is mapped onto the corresponding rotation of the result to its right – this rule is rotationally symmetrical, and these are all of the possible cases. Since each distinct initial state of a block is mapped onto a distinct final state, this rule is reversible. As will be shown later, the automaton corresponding to this rule is universal.

One could easily have written down an example of a rule that conserved 1's and 0's, but that didn't always map a distinct initial state of a block into a distinct final state – such a rule would be conservative, but not reversible. As the corresponding automaton evolved, it would forget all sorts of details about the initial state, but it would always remember the numbers of 1's and 0's.* Thus the existence of an interesting local conservation law does not depend on the rule being reversible!

In the language of digital logic, a gate from which it is possible to construct any boolean function of any number of input variables is a *universal* gate. If a logic gate is not universal, then no interconnection of such gates can be a computer. Thus the only candidates for universal CA's in the scheme described above are those whose rule corresponds to a universal logic gate.

In order to promote the CA rule of fig. 4 as a link between classical mechanics and computation, I will first discuss Fredkin's Billiard Ball Model (BBM) of computation[3] – a classical mechanical system that can be used to do digital computation. It will then be easier to discuss this rule, which I call the BBMCA – a purely digital model of computation which is closely related to the BBM.

7. The billiard ball model of computation

The BBM is a classical mechanical system, and obeys a continuous dynamics – positions and velocities, masses and times are all real variables. In order to make it perform a digital computation, we make use of the fact that integers are also real numbers. By suitably restricting the initial conditions we allow the system to have, and by only looking at the system at regularly spaced time intervals, we can make a continuous dynamics perform a digital process. In this case, we begin with a 2-dimensional gas of identical hard spheres. If the center of a sphere is present at a given point in space at a given point in time, we will say that there is a '1' there, otherwise there is a '0' there. The 1's can move from place to place, but their number never changes.

The key insight behind the BBM is this: *every place where a collision of finite-diameter hard spheres might occur can be viewed as a boolean logic gate***. What path a ball follows depends upon whether or not it hits anything – it makes a decision.

To see how to use this decision to do boolean logic, consider fig. 5. At points A and B and at time t_i, we either put balls at A, B, or both, or we put none. Any balls present are moving as indicated with a speed 's'. If balls are present at *both* A and B, then they will collide and follow the outer

* For each gate (block), we can tell after each step how many possible predecessors the result-block has. Thus we can count exactly how much information is lost at each step.

**The BBMCA models space as being uniformly filled wíth gates, and so a connection is already apparent.

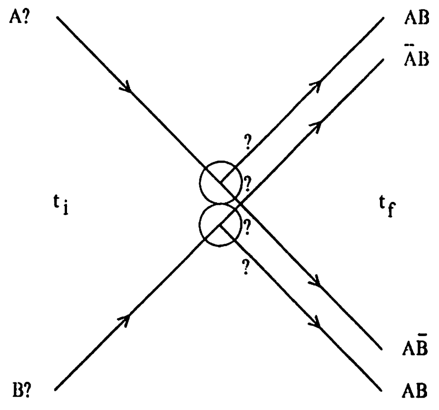


Fig. 5.

outgoing paths. Otherwise, only the inner outgoing paths will be used. At time $t = t_i$, position A is a 1 if a ball is there, and 0 otherwise (similarly for position B). At $t = t_f$, the four labeled spots have a ball or no ball – which they have is given by the logical function labeling the spot. For example, if $A = 1$ and $B = 0$, then the ball coming from A encounters no ball coming from B, and ends up at the point labeled "A and not B". A place where a collision might occur acts as a reversible, universal [3] 1-conserving logic-gate, with two inputs and four outputs. A path that may or may not contain balls acts as a signal-carrying wire. Mirrors (reflectors) allows bends in the paths. In order to be able to use the outputs from such a collision-gate as inputs to other such gates, we need to very precisely control the angle and timing of the collisions, as well as the relative speeds of the balls. We make this simple to do by severely restricting the allowed initial conditions. Each ball must start at a grid point of a Cartesian lattice, moving 'along' the grid in one of 4 allowed directions. See fig. 6. All balls move at the same speed. The time it takes a ball to move from one grid point to another we call our unit of time. The grid spacing is chosen so that balls collide while at grid-points. See fig. 7. All collisions are right angle collisions, so that one time-step after a collision, balls are still on the grid. Fixed mirrors are positioned so that balls hit them while at a grid point, and so stay on the grid. See fig. 8. By using mirrors, signals can be

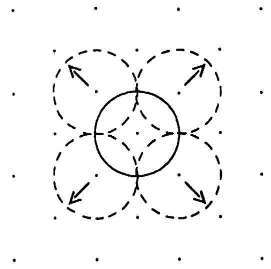


Fig. 6.

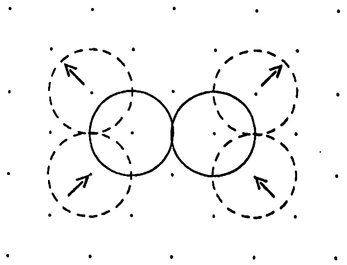


Fig. 7

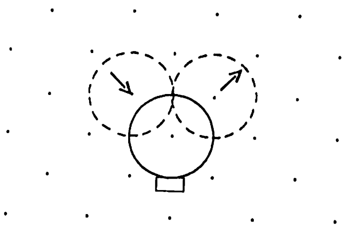


Fig. 8.

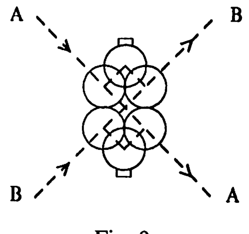


Fig. 9.

routed and delayed as required to perform digital logic. The configuration of mirrors in fig. 9 solves the problem of making two signals cross without affecting each other. (Notice that if two balls come in together, the signals cross but the balls don't!).

Mirrors and collisions determine the possible paths that signals may follow ('wires'). In order to ensure that all collisions will be right-angle collisions (and not head-on, for example, which would take us off our grid) we can label all 'wires' with arrows, and restrict initial conditions and interconnections so that a ball found on a given 'wire' always moves in the labeled direction.

Thus our universal gates can be connected as required to 'build' a computer. Computations can be pipelined – an efficient 'assembly-line' way of doing things, where questions flow in one end and finished products (answers) flow out the other, while all the stages in between are kept busy. Reversibility turns out not to be a great hindrance – unwanted intermediate results can be mostly 'erased' by copying the answer once you have it, and then running the computation backwards to get rid of everything but a copy of the inputs.

This then, in brief, is the BBM. Kinetic energy is conserved, since all collisions are elastic. Momentum is not conserved, since the mirrors are assumed to be fixed (infinitely massive).

8. The BBM cellular automaton

When viewed only at integer time-steps, the BBM consists of a Cartesian lattice of points, each of which may 'contain' a 0 or a 1, evolving according to a local rule. It would therefore seem to be a straightforward matter to find a CA rule that duplicates this digital time evolution.

Unfortunately, the most direct translation of the BBM into a CA has several problems. First of all, to have separate states of a cell to represent 4 kinds of balls (4 directions) an empty cell and a mirror, and to have the balls absolutely conserved (as they are in the original BBM) would require a standard "change the center cell" rule with 6 states per cell, and a 17 cell neighbourhood. Such a rule has a very large number of possible configurations for its neighbourhood, which makes it unwieldy. Moreover, many of these configurations involve such events as head-on collisions, which were disallowed in the BBM – a CA rule, however, should be defined for all configurations. It is not at all clear how to extend the BBM rule to these extra cases, and still have it remain reversible and 'energy' conserving.

At the expense of making collisions cause a slight delay, we can get away with the very simple rule of fig. 4, which involves only 2 states per cell in a 4 cell neighbourhood, is reversible, and conserves the number of ones (and zeros) in all cases. The $\begin{smallmatrix}□&□\\■&□\end{smallmatrix} \rightarrow \begin{smallmatrix}□&■\\□&□\end{smallmatrix}$ (and rotations) case in fig. 4 is the one that causes an isolated '1' to propagate in a straight line, in one of four directions (depending on which of the four corners of its starting block you put it in). See fig. 10. The legend "solid" or "dotted" below each of these automaton configurations tells you whether the grouping of cells into blocks for the next application of the rule

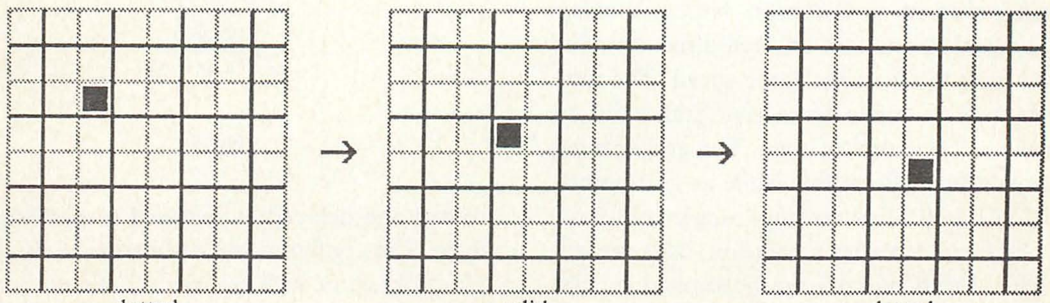


Fig. 10.

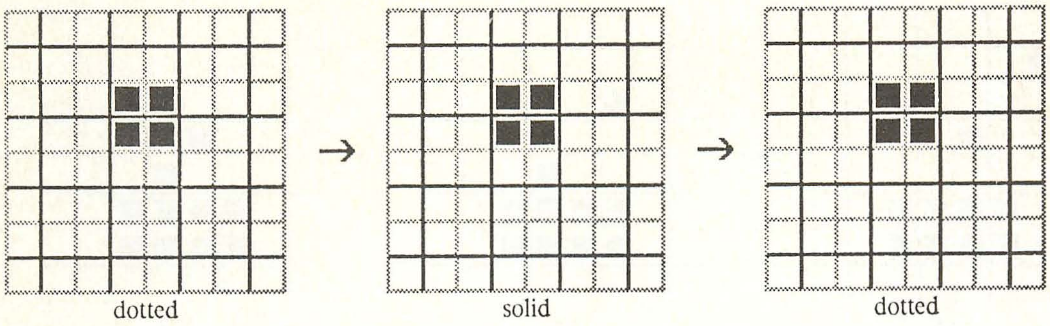


Fig. 11.

is indicated by the solid or the dotted lines. In the diagrams, a 0 is shown as an empty (blank) cell, a one is shown as a filled-in cell. Since $\begin{smallmatrix}\square & \blacksquare\\ \square & \blacksquare\end{smallmatrix} \longrightarrow \begin{smallmatrix}\square & \blacksquare\\ \square & \blacksquare\end{smallmatrix}$ (and rotations), a square of four ones straddling the boundary of two adjacent blocks will be stable–we will use such squares to construct mirrors. See fig 11. The four 1's straddle two dotted blocks horizontally, then two solid blocks vertically, and then two dotted again. Since $\begin{smallmatrix}\square & \blacksquare\\ \blacksquare & \square\end{smallmatrix} \longrightarrow \begin{smallmatrix}\blacksquare & \square\\ \square & \blacksquare\end{smallmatrix}$ (and rotations), pairs of travelling ones perform a billiard-ball type collision. See fig. 12. In all of these figures, the paths the ones were originally following have been lightly drawn in, to show that the 'and' case shown results in an outward displacement, just as in the BBM. (Unlike the BBM, there is a delay in such a collision, which we'll have to worry about in synchronizing signals). Finally, $\begin{smallmatrix}\blacksquare & \blacksquare\\ \square & \blacksquare\end{smallmatrix} \longrightarrow \begin{smallmatrix}\blacksquare & \blacksquare\\ \square & \blacksquare\end{smallmatrix}$ (and rotations) permits the reflection of double signals by a mirror. See fig. 13. The 'mirror' consists of two adjacent stable squares (notice that a square is stable no matter what you put next to it–its

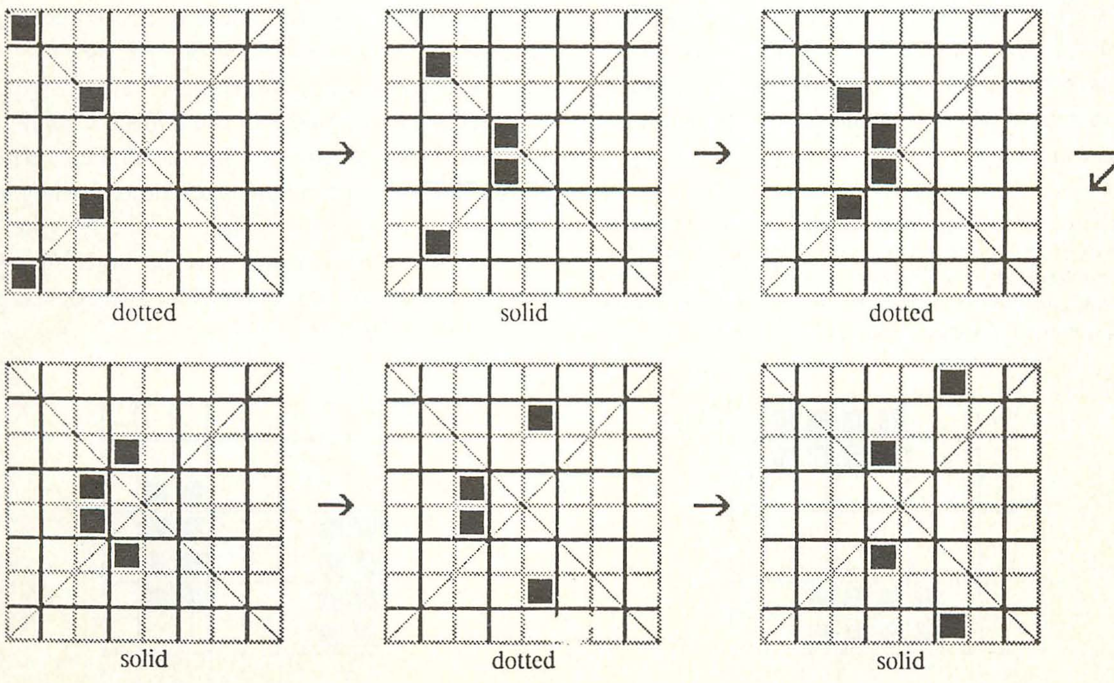


Fig. 12.

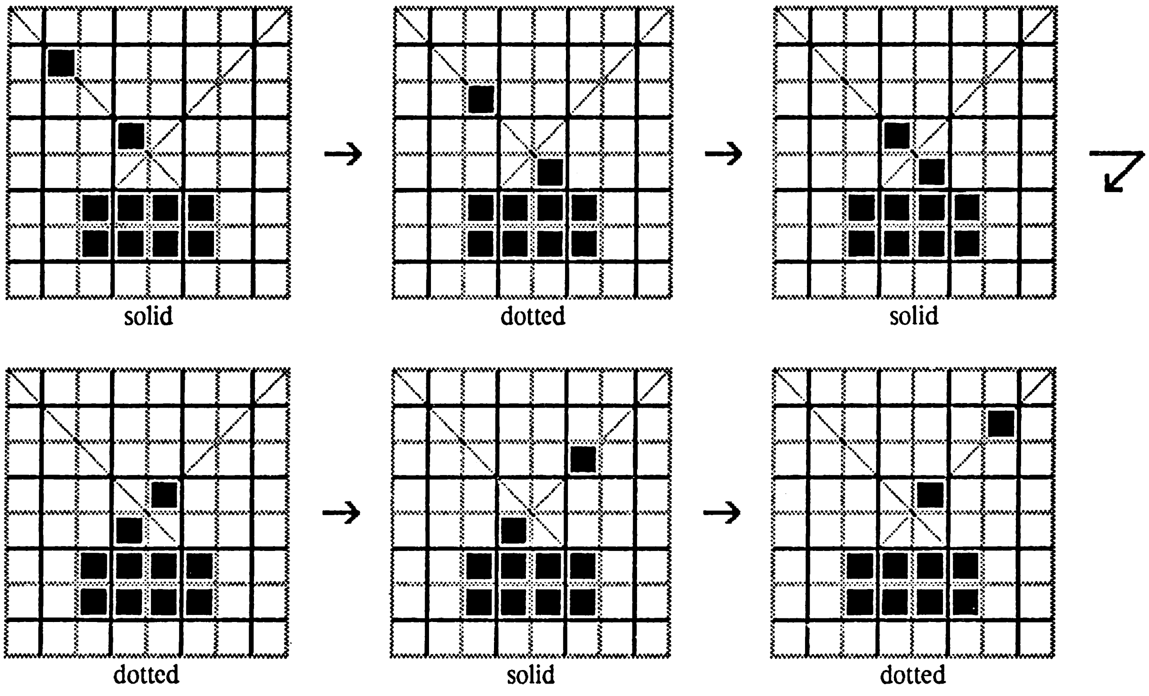


Fig. 13.

'decoupled'). Again, the signal path has been lightly drawn in. After each reflection such as that shown above, the signal has been delayed by a distance of one block along the plane of the mirror (in this picture, the signal winds up one block-column behind where it would have been had it not hit the mirror).

* We can tell how many steps a signal will take to traverse a given path (from one position where the signal is moving freely to another) by simply drawing the path joining the two points (including all points that may be visited by at least one '1') and counting how many cells are on the path.

In the BBM, such a reflection would cause no horizontal delay. We can compensate for such extra delays, as well as add any desired horizontal delay of 2 or more blocks. See fig. 14.

Suppose we want to arrange for two signals to collide, with the plane of the collision being horizontal. If we get the two signals aligned vertically and they are approaching each other as they move forward, they will collide properly. We can adjust the time it takes one or both signals to reach a given vertical column by using delays such as those in fig. 14*.

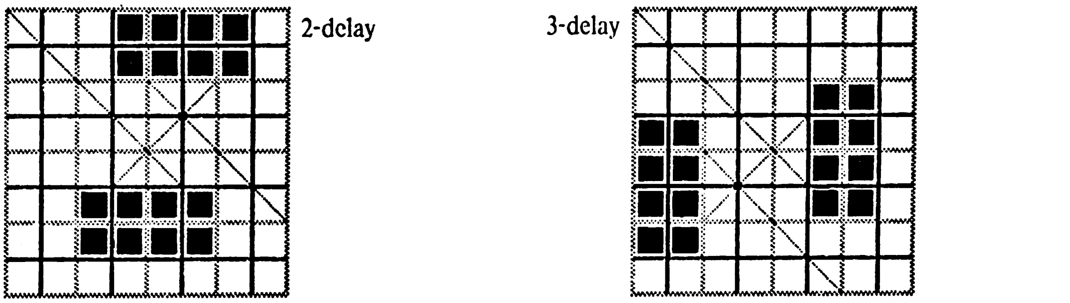


Fig. 14.

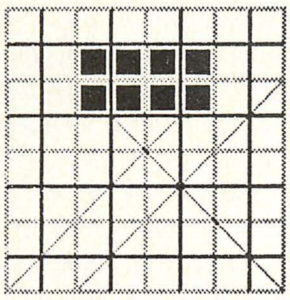


Fig. 15.

In order to allow signal-paths to cross without interacting, we use signal timing. By leaving a gap long enough for one signal (2 blocks) between all signals, we need only delay one of the paths by 2 blocks along the plane of the collision we're avoiding, in order to allow the signals to pass each other harmlessly. This gap is also enough to allow us to separate parallel output paths from a collision. See fig. 15. After the collision (fig. 12) the upper path already has a 1-block horizontal delay relative to the lower path. The mirror introduces a further 1-block delay, and so the upper signal passes through the timing-gap left in the lower signal path. With the addition of some extra synchronization and crossover delays, any BBM circuit can now be translated into a BBMCA circuit. Since the BBM has been shown to be a universal computer, the BBMCA is also.

There are many rules similar to the BBMCA that are also universal – for example, if we take the BBMCA rule of fig. 4 and modify it so that for each case shown, the result (right-hand side) is rotated 90-degrees clockwise on the 'dotted' steps, and counterclockwise on the 'solid' steps (i.e. $\begin{smallmatrix}■&□\\□&□\end{smallmatrix} \rightarrow \begin{smallmatrix}□&□\\■&□\end{smallmatrix}$ on dotted steps, and $\begin{smallmatrix}■&□\\□&□\end{smallmatrix} \rightarrow \begin{smallmatrix}□&■\\□&□\end{smallmatrix}$ on the solid steps, etc.) then we get another rule that is also computation universal. Its universality can be shown in a direct manner by using this rule to simulate the BBMCA (this rule can simulate a given BBMCA computation isomorphically using eight times as much space, and four times as much time)*.

*The idea for this BBMCA variation arose out of a discussion with Tommaso Toffoli.

9. Relationship of BBMCA to conservative logic

The collision-gate of fig. 5 has two inputs and four outputs. If we wish to consider it to be a conservative-logic gate (one that conserves both 0's and 1's) then we must regard it as a gate with four inputs and four outputs, two of the inputs being constrained to always be zeros.

The gate upon which the BBMCA is based also has four inputs and four outputs. Is there some connection here? Let us redraw the BBMCA rule in a different form (see fig. 16). The mapping of input variables onto output variables of the BBMCA has been redrawn as if the inputs all arrive and leave in a vertical column. If we use this correspondence to draw the four possible cases with $a = d = 0$, drawing □ for 0, ■ for 1, and showing each input/output case, we get an evolution (see fig. 17) which is logically the same as the collision gate. Thus the BBMCA rule of fig. 4 can be regarded as a completion of the collision-gate to a (reversible) conservative-logic gate!

```
                       a -|  |- A
ab → AB                b -|  |- B
cd   CD   becomes      c -|  |- C
                       d -|  |- D
```

Fig. 16.

```
□-|  |-□      □-|  |-□      □-|  |-□      □-|  |-■
□-|  |-□      ■-|  |-□      □-|  |-■      ■-|  |-□
□-|  |-□      □-|  |-■      ■-|  |-□      ■-|  |-□
□-|  |-□      □-|  |-□      □-|  |-□      □-|  |-■
```

Fig. 17.

10. Energy in the BBMCA

In the BBM, the kinetic energy is proportional to the number of moving 1's. In the BBMCA, if we let $p_{x,y,t\text{-}1/2} = c_{x,y,t} - c_{x,y,t-1}$, then $\Sigma_{xy}(p^2_{x,y,t\text{-}1/2}/2)$ counts the number of moving ones (each moving one disappears from one cell, and appears in another, so $\Sigma_{xy}p^2$ – which counts how many places change – would count each moving one twice).

The ones that aren't moving are at those places that were a one at $t-1$, and still are one at time t. Thus the number of stationary ones is $\Sigma_{xy}c_{x,y,t}c_{x,y,t-1}$. A complicated way of writing the (constant) total number of ones is

$$E_{t\text{-}1/2} = \sum_{xy} \frac{p^2_{x,y,t\text{-}1/2}}{2} + \sum_{xy} c_t c_{t-1}. \tag{9}$$

During a collision, some of the 'kinetic-energy' changes into 'potential-energy', and then it changes back again.*

Since (9) is a constant for *any* rule for which $\Sigma_{xy}c^2_{x,y,t}$ is constant, it is not possible to derive the particular rule from this expression. We might (for example) introduce the rule into (9) by using it to eliminate c_t (thus writing E as a function of $p_{t\text{-}1/2}$ and c_{t-1}) and see if we can push the mechanics analogy further.

Using number-of-ones to play the role of energy in BBMCA circuits and considering circuits for which we have only a statistical knowledge of what the different inputs will be, elaborate thermodynamic analogues can be established, but this will be discussed elsewhere. Although the overall system has a single deterministically evolving state, from the point of view of small pieces of the system, their inputs may appear random.

11. Conclusion

The laws of nature are the ultimate computing resource — the most efficient computation imaginable would make the most direct possible use of the physical interactions and degrees of freedom available. Physical quantities and concepts would have a direct computational interpretation. Computer scientists cannot hope to find the right quantities to use to talk about efficient computation until they have models of computation that are much closer to fundamental physics. Reversible Cellular Automata are offered as a step towards this end.

* One can think of mechanical models of the BBMCA for which the two terms of (9) are proportional to the physical kinetic and potential energy of the system midway between two steps.

Appendix A

A second-order, reversible, universal automaton

This appendix describes another BBM-type automaton. As before, we begin with a 2-dimensional cartesian lattice, this time with 3 states per cell, which we can designate as -1, 0, $+1$, and which we will draw as '\', blank, and '/' respectively in diagrams.

The time evolution will be given by $c_{x,y,t+1} = f(c_{\{x,y\},t}) - c_{x,y,t-1}$, where $f(c_{\{x,y\},t})$ is a function that 'looks' at the 3×3 neighbourhood with $c_{x,y}$ as its center cell, and '$-$' is taken mod3.

For each possible configuration of the neighbourhood, f will return a value of -1, 0, or $+1$. Just as head-on collisions never arise in BBM computations, many configurations of this RCA need not arise in order to 'build' a universal computer. We will leave these cases undefined – each choice for these undefined cases defines a distinct universal RCA.

An isolated '/' or '\' will correspond to a travelling billiard ball – if only the cases defined here arise, the number of such 'balls' will be conserved. An isolated '/' will propagate along a positively sloped diagonal – its evolution will be governed by the following cases:

```
000  000  /00  000  0/0  000  000  000
000  0/0  000  000  000  /00  00/  000
000  000  000  00/  000  000  000  0/0
```

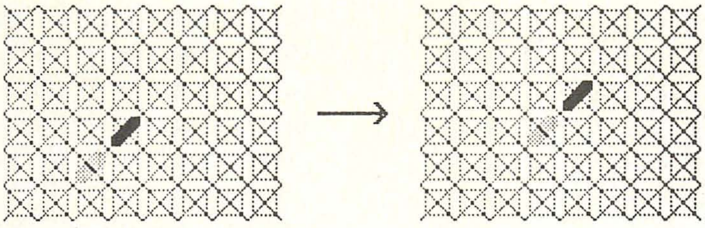


Fig. 18.

all return a '0' as the value for f;

```
00/  000
000  000
000  /00
```

both yield a value of '/' (i.e. + 1). A sample time evolution (using halftones to show a cell's contents at time $t-1$ and solid lines for time t, with diagonals lightly drawn through all cells) is shown in fig. 18. Intuitively, this rule at time t tries to make the '/' travel both forwards and backwards along its diagonal–subtracting away a '/' where it was at time $t-1$ just leaves a '/' in the forwards direction.

We define this rule to be rotationally symmetric. It will be helpful to adopt the following convention: the 90-degree clockwise rotation of

```
000          \00
000  → /  is 000  → \.
/00          000
```

Inversions are defined analogously. Thus an isolated '\' will follow a negatively sloped diagonal path if the propagation of signals is governed by the cases:

```
000  000  /00  0/0
000  0/0  000  000  → 0,
000  000  000  000
```

```
000
000  → /
/00
```

(and rotations and inversions).

For compactness in writing the complete rule, we adopt the convention that inversions as well as rotations of the cases given are mapped onto the corresponding inversions or rotations of the result given.

These cases become zero:

```
\\\ \\0 \\0 \\0 \\0 \\/ \\/ \0\ \0\ \0\ \00 \00 \00
000 000 /00 /00 //0 000 /0/ 000 /00 /00 000 /\0 /\/ ,
/// //0 000 00/ 00/ //\ /\\ /0/ 000 00/ /00 000 000
```

```
\00 \00 \00 \00 \00 \00 \00 \00 \00 \00 \00 \00 \00
/0\ /0\ /00 /00 /00 /00 /00 /00 /0/ /0/ //\ //0 //0 ’
000 00/ \00 \0/ 000 00/ 0/0 0// 000 00/ 000 000 00/
```

```
\0/ \0/ \0/ 00\\ 0\\ 0\0 0\0 0\0 0\0 00\ 00\ 00\ 00\
000 /0\ /0\ 000  000 000 000 000 /0/ 0\0 00\ 000 000 ’
/0\ \0/ 000 000  0// 000 00/ 0/0 0\0 000 00/ 000 00/
```

```
00\ 000 000 000 /\\ /\\ /0\
00/ 0\0 000 /0\ 000 /0/ 000
000 000 000 \0/ \// \\/ \0/
```

These cases become one:

```
\00 \00 \00 \00 \0/ \0/ \0/ \/\ 0\0  0\/ 0\/ 0\/
/\0 /00 /00 /00 /0\ /00 /00 /00 /\0  \00 00\ 000 ’
\/0 /\0 /00 /0/ 00/ 000 00/ 000 \/0  /00 000 /00
```

```
00\ 000 000 000 000 000 000
/\0 000 000 /\\ /\0 /\0 /\/
\/0 /00 /0/ \// \/0 \// \/0
```

(plus rotations and inversions). There are 2617 undefined cases.

Using this rule, a mirror is shown in fig. 19. We needed to define certain cases just to allow a mirror to remain unchanged when no signals are nearby.

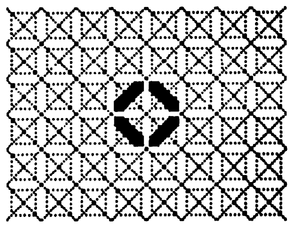


Fig. 19.

A signal bouncing on a mirror is shown in fig. 20. (Notice that there is no horizontal delay, as there was in the BBMCA). If this signal had been shifted one column to the right, it would have passed the mirror unaffected. We put some mirrors near places where signals might collide, so that (with its small neighbourhood) this rule can simulate an attractive collision – the signal paths will be displaced inward in a collision, rather than outward as in the BBM. See fig. 21. (If a signal arrives on

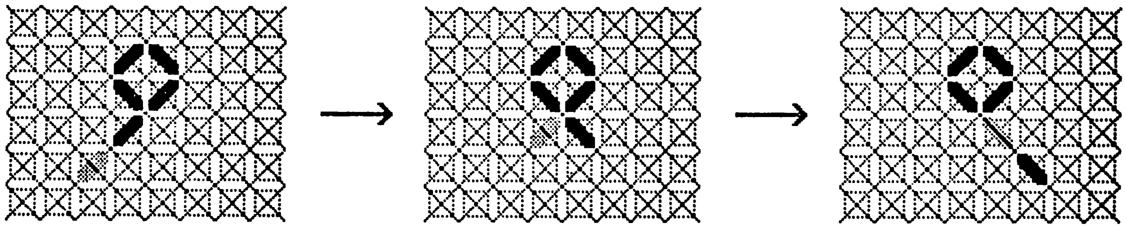


Fig. 20.

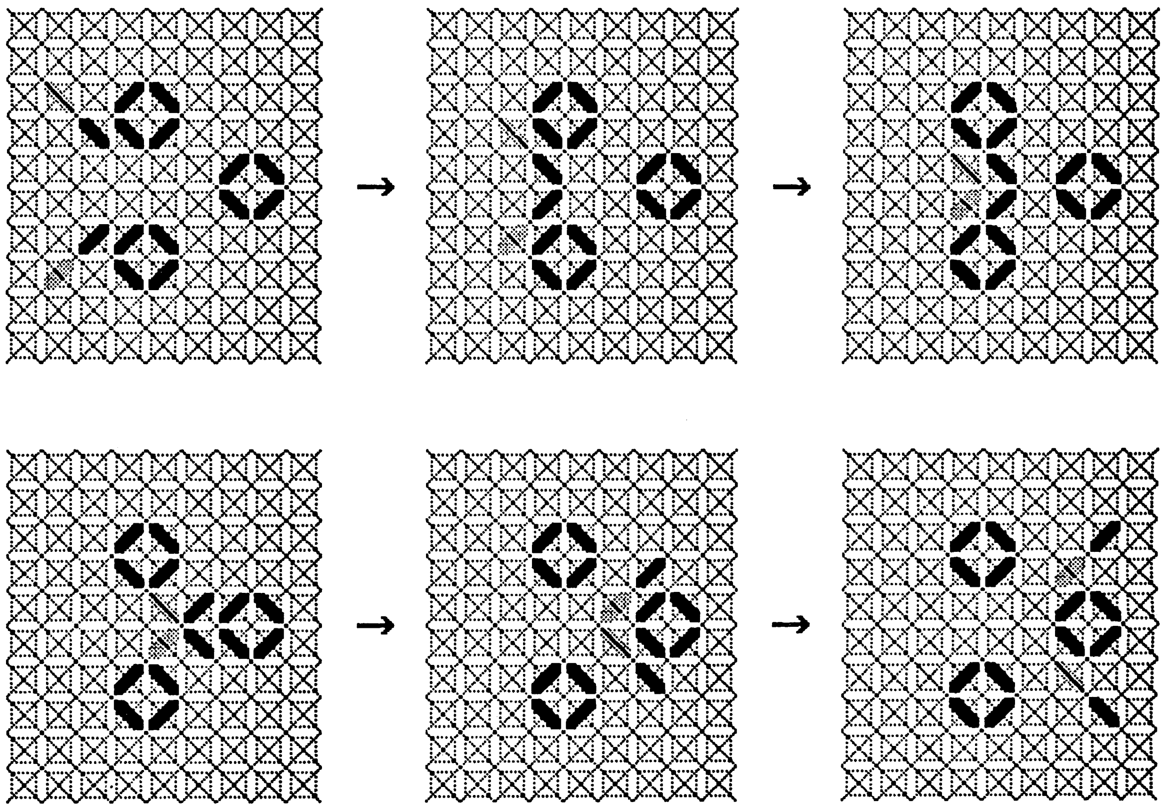


Fig. 21.

just one path, it goes through without any displacement). Two such gates, back to back, can be used to make signals cross over without affecting each other. See figs. 22 and 23.

Since all collisions occur without any delay along the plane of the collision, considerations of synchronization are very similar to those in the BBM. The proof of this automaton's universality is essentially the same as for the BBM.

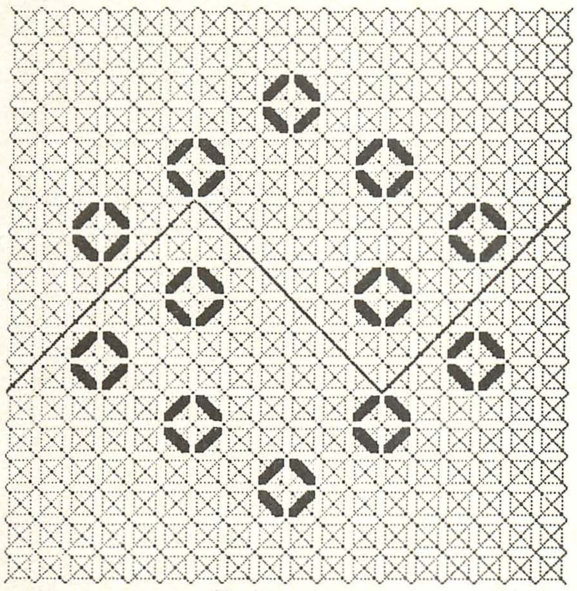

Single one case

Fig. 22.

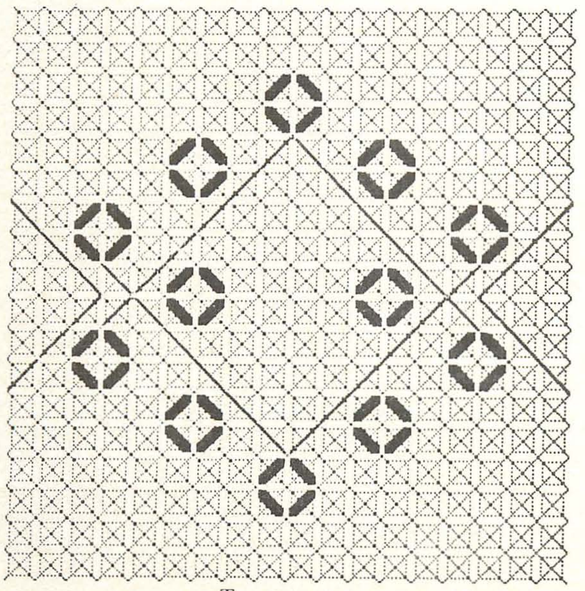

Two ones case

Fig. 23.

To give another example of a universal second-order RCA, we can begin with the BBMCA rule. If f_s is the global rule that applies to the solid blocking and changes an entire configuration into the next configuration, and similarly f_d applies to the dotted blocking, then we can describe the BBMCA evolution by

$$S_{t+1} + S_{t-1} = f(S_t), \tag{10}$$

where $S_{t+1} + S_{t-1}$ is taken to be the configuration obtained by taking the cell-by-cell sum (mod 2) of S_{t+1} and S_{t-1}, and $f(S_t) = f_s(S_t) + f_d(S_t)$ is also such a sum.

(10) can be rewritten in the form (4) with a 3×3 neighbourhood and a dependence on the parity of the center cell's position (parity of $x + y$). In a similar manner, *any* invertible CA rule can be written in the form (10), and in the form (4) if it is locally invertible.

References

[1] C.H. Bennet, "Logical reversibility of computation," IBM Journal of Research and Development **6** (1973) 85–91; "The Thermodynamics of Computation," Int. J. of Theo. Phys. **21** (1982) 905–940.

[2] E. Fredkin, private communication.

[3] E. Fredkin and T. Toffoli, "Conservative Logic," Int. J. of Theo. Phys. **21** (1982) 219–253.

[4] E. Berlekamp, J. Conway and R. Guy, "Winning Ways for your Mathematical Plays", vol. 2 (Academic Press, New York, 1982).

[5] M. Gardner, "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," Scientific American **223**:4 (1970) 120–123.

[6] R. Landauer, "Irreversibility and heat generation in the computing process," IBM Journal of Research and Development 5 (1961) 183–191.

[7] C. Shannon and W. Weaver, The Mathematical Theory of Communication (Univ. of Illinois Press, Illinois 1949).

[8] T. Toffoli, "Computation and construction universality of reversible cellular automata," Journal of computer systems science 15 (1977) 213–231.

[9] T. Toffoli, "CAM: A High-Performance Cellular-Automaton Machine," Physica 10D (1984) 195–204 (these proceedings).

[10] J. Von Neumann, Theory of Self-Reproducing Automata, (ed. and compiled by A.W. Burks) (Univ. Illinois Press, Illinois, 1966).

Home 5279045
MIT 253 3505
5893

Physica 10D (1984) 96–116
North-Holland, Amsterdam

SIMULATING PHYSICS WITH CELLULAR AUTOMATA*

Gérard Y. VICHNIAC
MIT Laboratory for Computer Science, Cambridge, Massachusetts 02139, USA

Cellular automata are dynamical systems where space, time, and variables are discrete. They are shown on two-dimensional examples to be capable of non-numerical simulations of physics. They are useful for faithful parallel processing of lattice models. At another level, they exhibit behaviours and illustrate concepts that are unmistakably physical, such as non-ergodicity and order parameters, frustration, relaxation to chaos through period doublings, a conspicuous arrow of time in reversible microscopic dynamics, causality and light-cone, and non-separability. In general, they constitute exactly computable models for complex phenomena and large-scale correlations that result from very simple short-range interactions. We study their space, time, and intrinsic symmetries and the corresponding conservation laws, with an emphasis on the conservation of information obeyed by reversible cellular automata.

1. Introduction

In 1948, von Neumann embarked on an ambitious project: to show that phenomena as complex as life – the survival, reproduction, and evolution of complex forms of organization – can be reduced in principle to the dynamics of many identical, very simple primitives capable of interacting and maintaining their identity. First, von Neumann considered the interaction of vortices and particles in suspension in some "primordial soup". Obviously, such a model was intractable, so, following a suggestion by Ulam, he adopted a fully discrete approach: space, time, and even the dynamical variables were defined to be discrete.

The resulting *cellular-automaton* theory describes a universe consisting of an homogeneous array of "cells". Each cell is endowed with a finite number of states, and evolves in discrete time according to a uniform local transition rule. The rule can be seen as a function whose arguments are the states at time t of the neighboring cells (and possibly the state of the considered cell itself) and whose value is the state of the considered cell at time $t+1$. The rule is uniform in that it is the same all over the array. Since all the cells "compute" their new state simultaneously, cellular automata are often seen as a paradigm of distributed computation [55]. Dyson [15] and Bernstein [4] have given lively accounts of the surprising success of von Neumann's enterprise [7], a self-reproducing cellular-automaton that anticipated the discovery of the duplicative function of DNA.

Further uses of cellular automata in biology have been numerous (for a bibliography, see Kaufmann [31]). The very name "Life" for the famous cellular automaton that John Conway invented in 1970 [21] still reflects the biology-motivated origin of cellular automata. The similarities between cellular-automaton behavior and that of many physical systems are also quite suggestive. In this paper, I shall examine how these similarities have and can be exploited in order to apply cellular automata to physics. (Other discussions can be found in these proceedings.)

There are several ways one can think of using cellular automata to simulate physics. I found it useful to distinguish between the three following approaches:

(i) Cellular automata as mere *computational tools*. This viewpoint is motivated by the following

*This work was supported for its major part by an IBM post-doctoral fellowship, and also by grants from DARPA (No. N00014-75-C-0661 and N00014-83-K-0125), and from NSF (No. 8214312-IST).

facts. In today's fastest general-purpose computers, signals spend more time and dissipate more heat in wires than in processors [28]. Furthermore, the latter become less expensive than the former. A computer architecture that aims to maximize the density of active elements at the expense of that of wires leads precisely to the cellular-automaton layout of identical processors*. For this reason, the existing and planned hardware cellular-automaton machines [56] are attractive parallel computational resources for many lattice models of physics. Moreover, in a cellular-automaton simulation of these models, the topology of the simulated object is reproduced in the simulating device. In contrast, in a general-purpose computer simulation of a lattice system, the data relative to sites i and $i+1$ have no particular reason to be stored in neighboring memories.

(ii) Cellular automata as fully discrete *dynamical systems*. In this approach, cellular automata are relevant to physics only insofar as dynamical systems are relevant to physics.

(iii) Cellular automata as original *models* for actual physical phenomena, possibly competing with existing continuum models.

These approaches are ranged in order of depth and ambition in the use of cellular automata to simulate physics: a purely ancillary role in the first one, a fundamental one in the third. The second category stands in an interesting intermediate position: discrete dynamical systems, in this view, attempt not to simulate specific physical phenomena but rather to embody general physical ideas**. A given cellular automaton defines its own discrete universe. It turns out that many of these cellular-automaton universes are inhabited by beings that are most often seen in the theoretical physicist's menagerie, such as symmetries and conservation laws, a conspicuous arrow of time in reversible microscopic dynamics, order parameters and nonergodicity, nonseparability, causality and light-cone, relaxation to chaos through period doublings, and, most instructively, the appearance of complex phenomena and large-scale correlations resulting merely from a very simple short-range interaction. This second approach aims at what Stan Ulam has wonderfully called "imaginary physics" as opposed to "real physics," the object of approaches (i) and (iii).

These three approaches have an important common feature that is unique to fully discrete systems. They provide a third alternative to the classical dichotomy between models that are solvable exactly (by analytical means) but are very stylized, and models that are more realistic but can be solved approximately (by numerical means) only. In fact cellular automata have enough expressive power to represent phenomena of arbitrary complexity, and at the same time they can be simulated exactly by concrete computational means: actual runs by a simulating device like CAM [56] are an exact implementation of the mathematics, and they are automatically free of numerical noise since discreteness is a defining component of the model itself. In other words, we have here a third class: that of *exactly computable* models. To appreciate the originality of this type of modeling, one should keep in mind that there is no attempt here to solve any given equation, in fact cellular automata do not engage in any numerical processing, they merely perform simple space-dependent logical decisions. In other words, this third class aims at digital *non-numerical simulations* of physical phenomena.

When faced with a classification like the one given above, a healthy reaction is to try to invalidate it with examples that cannot fit in any specific category, but seem to belong to all. An example that comes immediately to mind is the Ising model for magnets and binary alloys. In this model, the variables are defined at the sites of a regular lattice, they take only two values and interactions occur between neighboring sites only. Replace "lattice" by "array", "sites" by "cells"

* In that sense, a cellular automaton is nothing but an array-processor with a few bits rather than a whole floating-point variable at each site.

** Cellular automata have also been used, in this approach, to illustrate ideas from *chemistry*, viz., the study by Greenberg, Green and Hastings [24] of the reaction-diffusion equation.

and "iteration step" by "time step," and you have a cellular automaton! Surprisingly enough, this seemingly reasonable mapping gives terrible results when applied to the equilibrium properties of Ising spins. However, cellular automata can do a marvelous job if the mapping

$$(1 \text{ spin}, 1 \text{ iteration step}) \mapsto (1 \text{ cell}, 1 \text{ time step}) \tag{1.1}$$

is discarded, (and the emphasis is somewhat shifted from approach (iii) to the more pragmatic viewpoint (i)).

This paper is organized as follows. Conventions, symmetry properties, and mechanical analogies are exposed in the next section. Section 3 contains examples of the use of cellular automata as computational tools. The puzzling failure the intuitive mapping (1.1) in what seems to be the easiest simulation, i.e., that of Ising spins at equilibrium, deserves some analysis. Section 4 discusses carefully the role in this failure of the discreteness of time and of the variables. It shows that in a general manner this double discreteness prevents an extension to fully parallel processing of currently used serial methods (e.g., Monte Carlo sampling) for the computation of averages in equilibrium statistical mechanics. Sections 5 and 6 expose approaches (ii) and (iii) respectively. Finally, section 7 presents a tentative conclusion and some open questions.

2. Conventions. Symmetry properties and mechanical analogies

2.1. *Counting rules*

This paper is concerned with one- and two-dimensional cellular automata with two states per cell, called "0" and "1". Furthermore, we shall mostly limit our study to transition rules that merely count how many of the neighboring cells are in state "1", but do not care about the location of these cells. For cellular automata with n neighbors, there are only 2^{n+1} such "counting* rules," as opposed to 2^{2^n} possible rules. Naming the rules is an exercise subject to practical constraints: a name should be as explanatory as possible, short, free of exotic characters, and start with a letter in order to be a legal filename accepted by most computer systems. Here is a possible convention: the rule name is made of a letter followed by digits. The letter refers to the type of neighborhood and the digits to the numbers of ones in the neighborhood for which the rule returns a one. We shall use here predicate calculus, as expressed, to fix ideas, in the BASIC computer language. Let N, S, E, W, NE, NW, SE, SW (for the neighbors), and C (for the center cell) be the values (0 or 1) assumed by the cells in the neighborhood, and let Y, or C^{t+1}, be the new value of the center cell that the rule returns. Let us start with one dimension and let $D = E + W$ designate the number of ones among the left and right neighbors. Consider now for example the "mod 2" rule, that assigns the state zero if the two neighbors are in the same state and one otherwise. This rule can be defined by the BASIC statement

```
Y = (D = 1)
```

which is understood as follows**. In the brackets stands a predicate that can be either true (1) or false (0) according to whether or not $D = 1$. The whole statement assigns the truth value (0 or 1) of the predicate to Y, the state of the center cell at time $t + 1$. Now we shall name the rule "D1," a shorthand of the predicate. Similarly if now the center cell is included in the neighborhood ($T = E + W + C$), a new "mod 2" rule is defined as

```
Y = (T = 1   OR T = 3)
```

and called "T13". (Rules D1 and T13 have numbers 90 and 150, respectively, in Wolfram's convention [60].)

* These rules have been called "totalistic" by Wolfram [60].

** BASIC uses the same symbol for assignment and equality and does not distinguish between logical and numerical variables. The resulting notation is compact but hard to decipher. In good old FORTRAN IV the statement reads Y = 0; IF(D.EQ.1) Y = 1.

Looking now at the two-dimensional square lattice, we can count the cells in state one in a given neighborhood and introduce the variables

$$Q = N + S + E + W, \quad V = Q + C$$

and, including the corners,

$$H = Q + NW + NE + SE + SW \quad \text{and} \quad M = H + C\,.$$

The names of these variables are not totally arbitrary. The neighborhood consisting of the center cell and its nearest neighbors is often referred to in the literature on cellular automata [2] as the von Neumann neighborhood and as the Moore neighborhood when the next nearest neighbors are included. Hence the letters V and M (cf. also the Roman numeral meaning for V). The letters D, T, Q, and H stand for the initials of 2, 3, 4, and 8 in French (H is also the 8th letter of the alphabet.)

We shall see in the next section examples of two-dimensional cellular automata on the hexagonal lattice where each cell has six contiguous neighbors. Cellular-automaton rules on this lattice can in fact be simulated by rules in the Moore neighborhood by deleting two *opposite* corners, say NW and SE. Let X be the number of cells among the six neighbors that assume state one

$$X = N + NE + E + S + SW + W$$

and $Z = X + C$ that quantity when the center cell is included in the count. Notice that we can also obtain a tessellation of the square lattice with Y-shaped patterns and thus simulate the honeycomb lattice (on which each site has three immediate neighbors). We cannot, however, simulate cellular-automaton rules on this tessellation in a uniform way because we would need to remove an *odd* number of cells in the Moore neighborhood.

We shall also consider in section 5 rules that do not belong to the class of counting rules but to an immediate generalization, that includes rules where the values of Q and H for which $Y = 1$ depend on the state C of the center cell. "Life" is such a rule,

$$Y = ((H = 2 \quad \text{AND}\, C) \quad \text{OR} \quad (H = 3))\,,$$

which can also be written as

$$Y = ((C = 0 \quad \text{AND} \quad H = 3) \quad \text{OR}$$
$$(C = 1 \quad \text{AND} \quad (H = 2 \quad \text{OR} \quad H = 3)))$$

and denoted by the concatenation H3H23. Rules of this type can be called "double-counting," there are 2^{10} and 2^{18} of them in the von Neumann and Moore neighborhoods, respectively.

2.2 *Reversible rules*

A rule is said to be *reversible* if it is backward deterministic, i.e., if each configuration (or set of m configurations for rules that are mth order in time) has a unique predecessor. There is a simple way, due to Fredkin, to construct a wealth of reversible rules. Just take any rule f involving n states per cell, note the value it returns and subtract from it, in modulo n arithmetic, the value that the center cell assumed at time $t-1$:

$$C^{t+1} = f(\text{neighborhood at time } t) - C^{t-1} \pmod{n}\,. \tag{2.1}$$

This relation can be solved uniquely for C^{t-1}, even if f is not invertible, and this feature makes the new rule reversible. Notice that for $n = 2$, the subtraction modulo 2 is simply the exclusive OR Boolean operation. The new rule in now second order in time, but it can be made first order by including in the present the states of the past, i.e., by endowing each cell with n^2 states. In section 5 these reversible rules will be noted by appending an R to the names of the rules f of which they are made.

Notice that the rules constructed with (2.1) are not only reversible, they are also *time-reversal invariant*: a sequence of configurations can be obtained in reverse order with the *same* rule, simply by inverting the two last configurations.

Not all reversible rules are invariant under time reversal. For example, the rule where the center cell takes the state of its W neighbor (Y = W) generates a global shift toward the right. The past can be exactly recovered, not using the rule, however, but by (Y = E) instead. For the arguments concerned with the conservation of information, simple reversibility is all we need. For example, many reversible rules with special simple initial conditions evolve into what seems to be a totally random regime, despite the forward and backward determinism of the dynamics. The simplicity of the initial condition is of course still present in the obtained pseudo-chaos. It has only diffused into many-cell correlations*. Also, attractors are not allowed in the realm of reversible rules. These rules cannot exhibit the fascinating self-organizing behaviour discovered by Wolfram [60], since this effect requires *mergers* in their evolution. These are precisely the mergers that permit the non-reversible rules to circumvent the second principle of thermodynamics and create order from chaos, (a feat also performed by "Life"). On the other hand, when closer analogies with mechanics are the issue, time-reversal invariance is crucial. For instance, a rule like (Y = W) always shows "in what direction the time flows". But for time-reversal invariant rules, this is true only when the cellular automaton is "out of equilibrium and relaxes towards it"; once the system has thermalized, the conspicuous time-arrow has disappeared. It reappears, however, in the opposite direction, if two consecutive configurations are inverted, i.e., if "all the velocities are reversed," in an exact realization of Loschmidt's paradox [12].

Furthermore, the second-order system (2.1) makes two consecutive configurations act somewhat like a point in a phase-space, in which many of the results of classical mechanics still hold. This is for example the case of Liouville's theorem: the absence of mergers together with the discrete structure of the phase-space make the density of points invariant in a reversible evolution. This encourages further analogies. Like in mechanics, reversible rules have as many conserved quantities as variables that describe the initial conditions [34]. We also expect that some of these invariants are more important than others and assume that one of these is an "energy" that generates the evolution in a way Hamiltonians do in mechanics. For more results on "the statistical mechanics of cellular automata," see the detailed article of that title by Wolfram [60], and also section 5.1 of this paper.

2.3. *Growth inhibition: convex confinement and shape-confinement*

Many cellular-automaton patterns grow without limits out a single seed of one in a background of zeroes. For many other rules, on the other hand, patterns remain confined inside fixed boundaries. There are two types of such boundaries: they can be convex or they can have more general shapes. Let us concentrate first on rules in the von Neumann neighborhood, for which the convex shapes are rectangles. Some rules confine patterns to the smallest rectangle that encloses all the ones of the initial conditions. We can call such rules *convex-confined*, or rectangle-confined. If furthermore the ones do not even fill concavities, i.e., if they do not explore the convex islands and peninsulae of zeroes of thickness larger than one cell, the evolution can be said to be *shape-confined*. The contrast is manifest for initial conditions of ones forming hollow or L-shaped patterns. By a direct consequence of their definition, counting rules of the form $Qd_1, \ldots, d_n$ and double-counting rules of the form $Qd_1, \ldots, d_nQd'_1, \ldots, d'_{n'}$ are rectangle-confined if all the $d_1, \ldots, d_n$ are larger than one, and they are shape-invariant if these digits are larger than two. Likewise, in the Moore neighborhood, counting

* In "real physics," a diffusion into many-body correlations is irreversible, because of the macroscopic nature of possible measurements (with the exception of spin-echo experiments, that actually involve one-body operators only, see [40, 1] for lucid discussions). In cellular-automaton experiments, on the other hand, microscopic degrees of freedom can be accessed easily. In particular, the encoding of an initial condition can be extracted out of the many-cell correlations simply by applying the reverse rule, a striking illustration of the importance of the exact computability of cellular automata.

rules H$d_1, \ldots, d_n$ and double-counting rules H$d_1, \ldots, d_n$H$d'_1, \ldots, d'_{n'}$ are convex-confined if the $d_1, \ldots, d_n$ are all larger than four, and shape-invariant if they are larger than five. In the hexagonal lattice, these exclusive thresholds are three and four. These results hold also for the reversible extensions of these rules by (2.1).

Confined cellular automata lend themselves to an easier systematic study, their available space being finite by their own intrinsic laws, and not as a consequence of the necessary finiteness of the practical means of their simulation. Their available time is finite as well, of course: the finite number of cells, states per cell, and bits that encode the rule do not contain enough resources for an infinite non-periodic evolution. Using here again the mechanical analogy, we shall call the finite period the "Poincaré recurrence time" when the rule is time-reversal invariant.

2.4. *Intrinsic symmetry properties*

Some rules can be transformed into others by simple "intrinsic" symmetry operations that involve the states, rather than space and time. For example the value of rule T023 is the binary complement of that of T1, and we write this

$$\overline{\mathrm{T1}} = \mathrm{T023}\,.$$

If one asks now what rule T1 looks like when it is applied to the binary complement of a configuration, it looks like T(3 − 1), that is T2,

$$\overline{\mathrm{T1}(\overline{\text{neighborhood}})} = \mathrm{T2}(\text{neighborhood})\,.$$

We can call T2 the *conjugate* of T1. Some rules have special properties under conjugation: they can transform into themselves, or into their complement, but most rules are just like T1 and lack proper symmetry under conjugation. In the first case we call the rule *even* (or *invariant* or *self-conjugate*), and in the second case we call it *odd* under conjugation. Among the counting rules in the von Neumann neighborhood, for example, the even rules are Q2, Q04, Q024, Q13, Q123, Q0134, V05, V23, V14, V1234, V0145, and V0235; the odd rules being V012, V013, V024, V034, as well as their conjugate-complement V345, V245, V135 and V125. We shall see in section 5 that the reversible versions of some of these even rules have remarkable behaviours. These intrinsic symmetry properties can readily be extended to cellular automata with more than two states per cell.

3. Cellular automata as computational tools

3.1. *Dynamic and static models*

The available hardware resources [56] for the simulation of cellular automata can also find most natural applications to those areas of physics where the discretization of space, rather than being an artefact of a numerical simulation, is a feature of the physical system itself (e.g., crystals, for their lattice-vibration independent properties), or has already been made an integral part of an established theoretical model (e.g., lattice-gauge field theories and lattice-gas molecular dynamics). The models of statistical mechanics that involve a regular lattice with local interactions between sites are begging for cellular-automaton simulations. The analogy is particularly faithful for *dynamical* models. In fact, a wealth of numerical results in the study of percolation [14], nucleation, condensation, coagulation [6] and transport properties in molecular dynamics [49] are obtained by simulating on a general purpose computer the time evolution of systems that are actually instances of simple cellular automata or immediate generalizations of cellular automata*. Cellular-automaton machines are of course particularly appropriate for the treatment of these models (provided that the effects of the two other

* It is not surprising that the model chosen by condensed-matter physicists to challenge the astronomers' traditional view that long-range forces are necessary to account for the spectacular long-range order of spiral galaxies [50] is a simple generalization of cellular automaton.

discretizations – time and variables – are fully understood and tamed, see section 4). For *static* models, on the other hand, the correspondence is less immediate, because cellular automata deal with initial-value problems, whereas in the classical equilibrium statistical-mechanical theories, there is no natural microscopic dynamics, and one is primarily interested in averages over configurations of static thermodynamical quantities. To be sure, these quantities can be obtained by dynamic relaxation methods, thanks to the ergodic theorem, but cellular automata can also be useful even if one wishes to stick to the time-independent approach. In that case, cellular automata should reproduce the successive steps of some iterative method, where the automaton's "time" plays the role of the iteration index rather than the physical time. Cellular automata (with probabilistic transition rules) can update very efficiently the lattice configurations of Monte Carlo samplings (provided some traps are avoided, see section 4). They also could be a help in performing the bloc-spin transformations defined in the real-space renormalization group methods [43].

3.2. *Experiments with voting rules: percolation and nucleation, metastability vs. unstability, spinodals*

"Voting" rules*, i.e., counting rules that assign "0" or "1" according to the "popularity of these states in the neighborhood," yield genuine nucleation effects or relax to a percolating stable state. More precisely, rules that return a one if the count of that state among the neighbors is above a given threshold always lead to the growth of clusters of one of the "species" (0 or 1) until a stable configuration is reached. When they are applied to a random initial configuration, these voting rules behave in a way that depends extremely sharply on the initial concentration p of ones, we are then in presence of a *critical phenomenon*. The voting rules fall into two major classes. For a first category of them, cluster growth is limited for all the values of p; the clusters of the species in minority shrink, but the asymptotic configuration always contains surviving minority islands (that in some cases oscillate with period 2 – which is the highest possible period for voting rules [23]). Above a critical value p_c of their initial density, the ones percolate in the final stable configurations, i.e., they form a connected path that crosses the whole space. Under p_c, percolation occurs for the zeroes. The second major class contains convex confined rules (see section 2.3). During the first few steps, many 1-cells turn to "0", being surrounded by too few 1's. But local high-density fluctuations in the initial distribution of ones initiate the growth of clusters of 1's. The growth consists in filling concavities, and halts once the convex shapes are reached. But if two clusters touch (or almost touch) they create new concavities and the growth goes on and forms a larger single convex cluster. For $p > p_c$, the clusters merge and the whole array turns to "1" after a small number of steps there is nucleation. For $p < p_c$, the clusters stop growing before they can meet. They remain separated by a sea of 0's. To be sure, on an infinite lattice, this 0-phase is metastable: an exceptional fluctuation can create a very large cluster that will grow for ever, feeding on isolated small clusters. In the thermodynamic limit the fate of the array does not depend on p; yet this equilibrium configuration of all 1's can be reached through two very different dynamical regimes: the 0-sea is *unstable* for $p > p_c$, but the region $0 < p < p_c$ corresponds to a *metastable* coexistence line between the 0- and the 1-phase, the value p_c playing the role of a *spinodal point*. See color plates in Toffoli [56], this volume. In fact this classification of voting rules according to their percolating or nucleating nature corresponds also to an important structural difference.

A simple-majority vote is possible only for an *odd* number of voting cells, since this leads to an even number of cases, ranging from an unanimous "0" vote to an unanimous "1" vote. For example V345 (also called VGE3, to show that the count of one states is *greater or equal* to 3 among the 5 neighbors) is balanced between the three cases

* These rules are instances of "threshold functions" used in the theory of neural networks [31, 37, 51, 23, 54].

V = 0, 1, 2 and the three cases V = 3, 4, 5. In that sense Z4567 (alias ZGE4) and M56789 (alias MGE5) are also simple-majority rules. It turns out that these are the rules that yield percolation. Due to the 0–1 symmetry of the problem, or to the fact that simple-majority rules are also self-conjugate (see section 2.4) the critical concentration must be $\frac{1}{2}$. Voting rules with an even number of voting cells lead to a necessary bias: the mid-point cases (Q = 2, X = 3, and H = 4) must be assigned to zero or one (we exclude here tie votes, i.e., special prescriptions for the mid-point cases, and shall examine them in section 4 in the context of the Ising model). Such biases can be seen as external magnetic fields in the "0"- or "1-direction": an oscillating motion of the domain walls is observed when the direction of the bias is switched, e.g., when Q34 and Q234 are applied by intermittence. A bias can alternately be incuded by changing simple-majority rules and shift the threshold as in V2345 (alias VGE2). These biased voting rules yield nucleation with finite critical concentration. A second shift, as in V12345 (alias VGE1) removes the convex confinement and leads to nucleation even for an infinitesimal initial concentration of ones ($p_c = 0^+$). Values of the critical concentration of ones for the voting rules are given below (the fractions (m/n) indicate that the corresponding rules involve an inclusive threshold of m ones among n neighbors).

von Neumann neighborhood

QGE2	(2/4)	$p_c = 0.133$	(nucleation),
QGE1	(1/4)	$p_c = 0^+$	(nucleation),
VGE3	(3/5)	$p_c = \frac{1}{2}$	(percolation),
VGE2	(2/5)	$p_c = 0.0822$	(nucleation),
VGE1	(1/5)	$p_c = 0^+$	(nucleation);

Moore neighborhood

HGE4	(4/8)	$p_c = 0.333$	(nucleation),
HGE3	(3/8)	$p_c = 0^+$	(nucleation),
MGE5	(5/9)	$p_c = \frac{1}{2}$	(percolation),
MGE4	(4/9)	$p_c = 0.250$	(nucleation),
MGE3	(3/9)	$p_c = 0^+$	(nucleation);

hexagonal lattice

XGE3	(3/6)	$p_c = 0.266$	(nucleation),
XGE2	(2/6)	$p_c = 0^+$	(nucleation),
ZGE4	(4/7)	$p_c = \frac{1}{2}$	(percolation),
ZGE3	(3/7)	$p_c = 0.191$	(nucleation),
ZGE2	(2/7)	$p_c = 0^+$	(nucleation).

Obviously, the conjugates of the complements of these rules yield nucleation of zeroes for the same critical concentrations of zeroes. These numerical values were obtained using Toffoli's Cellular-Automaton Machine (CAM) [56], a simulator of size 256 × 256. The standard deviation (1/256) is small enough and allows a fast and accurate determination of the critical concentrations.

Variations on the voting rules can be introduced. Voting rules in the von Neumann neighborhood have other interesting sets of initial configurations besides the random one. These are made by "doping" at random sites a very small amount of zero- and one- "impurities" on a matrix consisting of a checkerboard pattern of zeroes and ones. This background is doubly "grey" because its states oscillate between "0" and "1" both in space and time. We see then the dynamics of two species (the clusters of zeroes and ones) on a grey background. This is in contrast with most other cellular automata with two states per cell where, as in "Life," patterns of ones evolve on a background of zeroes.

Another way to modify the voting rules is to complement the ballot of some of the voters. For example this method can introduce anisotropies by counting, in the von Neumann neighborhood, the N and S neighbors with a "sign" opposed to that of E and W. This leads to metamagnetic structures, that are ferromagnetic in one direction and anti-ferromagnetic in the perpendicular direction*. A similar procedure can be applied in the Moore neighborhood, giving a rule that aligns the future state of the center cell to those of the nearest neighbors but opposes the states of the next nearest neighbors. This rule corresponds to *frustrated*

* These structures are of particular interest because they display in 3 dimensions tricritical [26] and tetracritical [27] points.

bonds [58]: after some transient time the cellular automaton keeps on oscillating between the two ground states. In the next section we shall see similar oscillations that are pure cellular-automaton artefacts and do not correspond to any physical frustration.

4. Simulating the Ising model with cellular automata

4.1. *Motivations and main result*

It is natural to ask how well cellular automata simulate the Ising model if only because of the strong resemblances between these two systems. A second compelling motivation to investigate the Ising model is the recent intense interest in constructing special purpose machines dedicated to its simulation. These machines [47, 30] are based on a pipeline architecture and therefore involve serial processing at the bottom level. But the expected availability of cheap VLSI circuits [57] forces us to ask whether it is possible to simulate the wandering of Ising configurations through the canonical ensemble with a simultaneous updating of *all* the spins at each iteration. The answer to this question is negative. A simultaneous updating of two contiguous spins necessarily yields wrong results, and the mapping (1.1) must be discarded for this problem. We can instead endow each cell with four states in order to accommodate for *two* spins, or equivalently, perform an alternate updating of every second spin in a checkerboard pattern [11]. The practical consequences of the invalidity of the intuitive mapping (1.1) are benign: the maximal speed of a computation is still a half of what it would be if full parallelism were allowed. However, the theoretical import of this bound is surprising and reveals an irreducible serial constitutive feature in any systematic sampling of configurations in equilibrium statistical mechanics for discrete systems. In other words, the wandering of configurations does not correspond to any obvious cellular-automaton dynamics, despite what one might intuitively expect following the approach (iii) mentioned in the introduction.

4.2. *Global energies and local measurements*

The Ising model for magnets and binary alloys involve a regular lattice on each site i of which is a spin σ_i. The spins are the only variables of the model, they can take two values (up and down) or $(+1$ and $-1)$ and interact with their nearest neighbors only, via the Hamiltonian

$$H = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j , \tag{4.1}$$

where the sum is taken over neighboring pairs only. If $J > 0$, the spins tend to be parallel and there is a ferromagnetic phase below the critical temperature T_c. When the system is in equilibrium with a reservoir at temperature T, the mean value of an observable A in the canonical ensemble is given by

$$\langle A \rangle_T = \frac{\sum_{\{\sigma_i\}} A(\{\sigma_i\})\, \mathrm{e}^{-H(\{\sigma_i\})/kT}}{Z(T)} , \tag{4.2}$$

where the sum is taken over the set of all possible configurations of spins, k is the Boltzmann constant and the normalization denominator is the partition function

$$Z(T) = \sum_{\{\sigma_i\}} \mathrm{e}^{-H(\{\sigma_i\})/kT} . \tag{4.3}$$

The Monte Carlo approximation reduces the enormous number of terms in (4.2) to an "importance sampling" [42] of the configurations according to their total energy, given by (4.1). In principle, each iteration should include the computation of the r.h.s. of (4.1) in order to obtain E^t, the global energy of the configuration of step t, but a simple trick permits the economy of this computation. The trick consists in flipping one spin only at each step. This enables a simple *local* observation of the energy change $\Delta\epsilon_i$ at site i to yield the *global* energy increment $E^{t+1} - E^t$ (the quantity that actually enters the sampling criterion),

$$\Delta\epsilon_i = E^{t+1} - E^t . \tag{4.4}$$

(Measurements of the local energy changes also suffice for the multi-spin-coding technique [11], where several *noncontiguous* spins are updated at the same time, since the mechanism of this method can be simulated exactly by a standard serial scanning.) Consider now a Monte Carlo computation on a two-dimensional square lattice. With an orderly sampling of the spins, starting from the NW corner of the lattice and sweeping towards the SE, the energy change $\Delta\epsilon_i$ at site i involve spins that have already been updated as well as "old spins" that are contemporaries of the considered spin. Specifically,

$$\mathrm{C}^{t+1} = f(\mathrm{N}^{t+1}, \mathrm{W}^{t+1}, \mathrm{C}^t, \mathrm{E}^t, \mathrm{S}^t; r(\mathrm{C}^t), T) , \tag{4.5}$$

where $r(\mathrm{C}^t)$ is a random number updated at each site. (In the multi-spin-coding technique, E and S are also taken at $t + 1$). We use here the notation of section 2 but now of course the variables take their values in the set $\{\uparrow, \downarrow\}$ instead of the set $\{0, 1\}$. Notice also that in (4.5) $t + 1$ means after an entire sweep over the lattice, not after an elementary step, as in (4.4).

4.3. *The feedback catastrophe*

The parallel processing of all spins can be readily simulated with standard serial computational resources. Just write a Monte Carlo code, and modify it in order to make $\Delta\epsilon_i$ depend now on the old spins only:

$$\mathrm{C}^{t+1} = f(\mathrm{N}^t, \mathrm{W}^t, \mathrm{C}^t, \mathrm{E}^t, \mathrm{S}^t, r(\mathrm{C}^t), T) . \tag{4.6}$$

One then observes the following catastrophe, starting with an aligned configuration at, say, $T = 0.8T_c$ and a ferromagnetic coupling $J > 0$. During the first sweeps, some spin and flip back, very much like in a standard Monte Carlo calculation, but as soon as two spins (or cells) with contiguous corners flip during the same sweep, a spurious checkerboard pattern starts to grow, leading eventually to an oscillation between the states of *maximum* energy i.e., the two ground states of an antiferromagnet.* The stability of the oscillating monster is evident from (4.6). Each spin up, surrounded by four spins down, will flip in order to align with its neighbors, which themselves will also flip, doing "the same reasoning." This occurs whenever f stands for the Metropolis or the "heat bath" algorithm [5]. One might ask whether there exists a transition rule f, not necessarily related to the Monte Carlo practice, that would avoid the feedback and yet give the correct averages (4.2) for a very large number of iterations.

* A similar computer experiment has recently been reported by B. Hayes, Scientific American, October 1983.

4.4. *An argument against full parallelism*

We shall show now that there is no such rule. For simplicity, let us consider the one-dimensional case. What we are after is in fact a probabilistic cellular automaton. The look-up table for a probabilistic cellular automaton does not return a new state C^{t+1} but rather a random value. It takes the general form

$$\{(\uparrow,\uparrow,\uparrow),(\uparrow,\uparrow,\downarrow),\ldots,(\downarrow,\downarrow,\downarrow)\}\mapsto\{a(T),b(T),\ldots,h(T)\} .$$

This maps the eight possible neighborhood values to eight random variables. In the case $(\uparrow,\uparrow,\uparrow)$, for example, the center spin will be up with probability $a(T)$ and down with probability $1 - a(T)$, and similarly for the other neighborhood cases.

Let us consider now a lattice of N spins all in the up state

$$\cdots\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\cdots,$$

along with the very large set of all configurations of a given magnetization M, say $M = 2/3$. In these configurations the up spins are twice as numerous the down spins:

$$\cdots\uparrow\uparrow\downarrow\uparrow\uparrow\downarrow\uparrow\uparrow\downarrow\uparrow\uparrow\downarrow\cdots,$$
$$\cdots$$
$$\cdots\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow\uparrow\uparrow\downarrow\cdots,$$
$$\cdots$$
$$\cdots\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow\downarrow\cdots,$$

The configurations of this set have a very large range of excitation energies. Nevertheless, at a temperature such that $a(T) = 2/3$, they are all populated from the ground state with the *same* transition probability W,

$$W = a(T)^{2N/3}(1 - a(T))^{N/3}, \tag{4.7}$$

regardless of the particular value of their total energy. A probabilistic cellular-automaton rule dictates how many spins will flip, but not how many domain walls will be formed. In other words, even if a transition rule can "know" the energy cost of flipping one spin, it has no way to determine the global energy of the obtained configuration. Therefore these cellular-automaton transitions cannot conserve the Boltzmann distribution, a necessary condition [59] to sample the canonical ensemble at equilibrium*. The argument is quite general, it does not depend on our particular choice of temperature and configurations. The cellular-automaton transitions will not populate configurations according to their Boltzmann factor, which depends on the global energy. This is again because the transition rules have no access to that global energy, but only to the local energy changes, and when contiguous spins are updated simultaneously, the relation (4.4) ceases to be valid, (essentially because energy is a two-body operator**, by (4.1)).

4.5. *Role of discreteness*

The Ising spins take only two values. This clearly plays a role in the fact that the feedback oscillations are not damped. The only way for a spin not to be up is to be down; the model does not have room for "almost-up" spins. Similarly, one suspects that the discreteness of time has also an effect in this behaviour. This in fact can be studied in a quantitative way with the help of a model continuous both in time and in the values of the spins, where, so to speak, discreteness can be continuously introduced†.

Let us rewrite (4.3) in the form

$$Z(T) = \int \prod_i \mathrm{d}\sigma_i \delta(\sigma_i^2 - 1)\, \mathrm{e}^{-H(\{\sigma_i\})/kT}. \tag{4.8}$$

The delta function guarantees that despite the continuous notation, this expression is equivalent to (4.3). Adapting an idea of Parisi [45, 20] we can "soften" the spins and obtain an equation of motion for them. First we use the representation

$$\delta(\sigma_i^2 - 1) = \lim_{\lambda\to\infty} \lambda \sqrt{\frac{\pi}{2}}\, \mathrm{e}^{-(\lambda/4)(\sigma_i^2 - 1)^2} \tag{4.9}$$

of the delta function as a double Gaussian extremely peaked at $\sigma_i = \pm 1$, but refrain from taking the limit $\lambda \to \infty$. The parameter λ that characterizes both the height and the narrowness of the Gaussian controls to what extent our model is "two states per site." Furthermore, the continuous formalism allows some dynamics in the form of Langevin's equations of motion

* The reader familiar with reasonings based on the principle of detailed balance [5] might like first to reduce the eight random functions $a(T), \ldots, h(T)$ to only three independent transition amplitudes (using the symmetries of (4.1)); and then to construct a set of pairs of configurations for which micro-reversibility yields constraints that cannot be simultaneously satisfied by the three amplitudes. This is a straightforward task, too tedious though to be reproduced here.

** This argument alone is not enough. One should also prove the non-existence of sum rules that would express the sum over the spins of a two-body operator in terms of one-body operators (e.g., the virial theorem in mechanics). This is in effect what has been done above.

†I am thankful to Henri Orland for suggesting the following analysis and for writing a computer code to check its validity.

$$\frac{d\sigma_i(t)}{dt} = -\Gamma \frac{\delta H(\{\sigma_i(t)\})}{\delta\sigma_i(t)} + \eta_i(t) , \qquad (4.10)$$

where each spin is made a function of time, Γ is the inverse of a relaxation time, and $\eta_i(t)$ is a Gaussian random variable:

$$\langle \eta_i(t) \rangle = 0 , \quad \langle \eta_i(t)\eta_j(t') \rangle = 2\Gamma\delta_{ij}\delta(t-t') . \qquad (4.11)$$

To be sure, this is not the only possible dynamics for (4.8), but it reproduces the wandering through the canonical ensemble at $t \gg \Gamma^{-1}$ [25]. We now discretize the time according to the step Δt and use the simplest two-point formula for the time-derivative, and when we plug (4.1) and (4.9) into (4.8), eq. (4.10) takes now the form

$$\sigma_i(t+\Delta t) = \sigma_i(t) - \Delta t \Gamma \left(\lambda(\sigma_i^3(t) - \sigma_i(t)) + \frac{J}{T}\sum_j \sigma_j(t) \right) + \sqrt{\Delta t \Gamma}\,\eta_i(t) . \qquad (4.12)$$

It should be stressed here that this algorithm does not converge exactly to the Boltzmann distribution when Δt is finite. This is not the only problem with (4.12). Indeed, from the result of section 4.4, we can predict that this algorithm is numerically unstable for very large values of λ, because then (4.12) becomes in fact a cellular-automaton rule for the two-state per cell Ising model! Notice, however, that in this expression λ is multiplied by Δt, thus at vanishing time increments the system is stable. In a very general way, as Olivier Martin pointed out to me, the feedback catastrophe cannot occur in the continuous time limit, because the transition probabilities vanish with Δt.

The numerical meaning of the parameters in (4.12) can be assessed by means of the following estimates. The parameter λ constrains the value of each spin around $+1$ or -1, with a Gaussian effective potential of depth $\Gamma\lambda$, and the thermal noise capable of moving a spin from the well centered around $+1$ to that centered at -1 is of strength $\sqrt{2\Gamma/\Delta t}$. In the high T limit, where we can neglect the Hamiltonian term in σ_j, a spin can go from one well to the other and flip if $\sqrt{1/\Delta t}$ is as large as λ. (As both the constraint and the noise are Gaussian, it is not surprising that this estimate turns out to be well verified). In the low T limit, where the Hamiltonian term dominates, the probability for a spin to flip in the minimal number of time steps is w^{g-1}, where g is the number of states per spin (i.e., the number of values that a spin can assume on the numerical mesh) and w is the transition strength, at a given T, between two successive spin states. At very low T, w is close to unity, and we should not be astonished to see a catastrophe for $g = 2$! For a value of g as low as 16 (CAM can accommodate 256 states), the feedback seems fully damped. We have checked this by observing that the solution of (4.12) behaves in a similar manner when σ is constrained to take one of 16 values and when it has a floating-point computer representation.

4.6. *Temperature zero and optimization*

The difficulties encountered in section 4 reduce to the fact that global thermal equilibrium at finite temperature over the whole array cannot be obtained by a cellular-automaton local transition rule that follows the mapping (1.1). When, on the other hand, the concern is the ground-state properties of a system ($T = 0$), we have in fact a minimization problem and cellular automata based on (1.1) can prove to be very effective. Take the Ising case again. A cellular-automaton rule will produce the unwanted feedback only at sites of high local energy ϵ_i, but it behaves in a satisfactory way inside homogeneous domains. Define now a new rule that follows the cellular-automaton Ising rule for aligned domains, but ignores it in the checkerboard domains of high energy. The new rule will flip a coin instead. Physically, it puts these regions at an infinite temperature. We observe then a growth of the $T = 0$ domains at the expense of the dislocations and the oscillatory monsters. What was a weakness for finite temperatures, i.e., the failure to satisfy the zeroth principle of thermodynamics and obtain a uniform temperature over the array proves to be a strength at $T = 0$. The nucleation of cold domains is enhanced while the dislocations are locally burned. This method also gives spectacular results for the fast production of a single metamagnetic crystal (see the end of section 3).

The most interesting zero-temperature problems might well be the most difficult ones, that for example involve frustrated bonds. The standard procedure [32, 33] for the numerical investigation of the ground states of such lattices is by the simulation of a very gradual scheduled annealing with a careful Monte Carlo thermalization at every step down in T. If the cellular-automaton method could be adapted to this problem, it would be substantially faster than the standard method since it starts to breed the $T = 0$ domains from the very beginning of the calculation. To be sure, such an adaptation could turn to be a very difficult challenge, since in presence of frustration there is no evident local signature of the "good" domains.

5. Cellular automata as discrete dynamical systems

5.1. *Non-ergodicity and order parameters*

The most remarkable two-state-per-cell counting rule in the von Neumann neighborhood is, in my opinion, the Q2R rule. (It is the reversible rule constructed on the self-conjugate rule Q2 by Formula (2.1)). Notice that Q2 is a second way (besides Q13) to generalize to two dimensions the "mod 2" rule D1: it assigns a one whenever zeroes and ones are in equal number among the four neighbors). The new state Y of the center cell that Q2R defines can be expressed with the BASIC statement* (where CPAST = C^{t-1})

$$Y = ((Q = 2) \neq \text{CPAST}).$$

By construction, Q2R is time-reversal invariant and also rectangle-confined for patterns of ones in a background of zeroes. It therefore inherits all of the nice properties described in sections 2.2 and 2.3. The specific behaviour of Q2R can be described as follows: when the initial condition contains solid shapes of ones at $t = 0$ and $t = 1$, one always observes a rapid drop in the number of ones. This drop is however limited: the system must keep enough ones in order to accommodate for the encoding of the initial condition (it must "remember it"). After the transient drop, the conspicuous arrow of time disappears and the systems "thermalizes" at a nearly constant population of ones, typically of 1% of the original population. Moreover the motion of the remaining ones is most often limited to very small oscillations around fixed patterns. The dynamics of Q2R perform two minimization processes: that of the number of ones, and that of their motion. Also, the equilibrium patterns often form disconnected islands that from time to time grow and exchange information. This fact and the frequent formation of genuine clocks and counters suggest that Q2R is capable of universal computation (i.e., it can simulate, for special initial conditions, the operations of a general-purpose computer).

In view of section 2.2, we shall use freely the terminology of classical mechanics. The first minimization process encourages us to consider the number of ones as a kind of potential energy. The second one indicates that the potential wells can be deep and narrow. The kinetic energy could be related to the number of zeroes created in the original rectangle: it is zero at $t = 1$, when the potential energy is maximum. Yet, the exact form of the analogue of the total energy is not evident. Nevertheless we know that this total energy is conserved, and we can ask whether or not an evolution eventually visits all the configurations of that energy, rendering averages over the energy surface in the phase-space equivalent to averages over very long times. In other words, we ask if the rule Q2R is *ergodic*. The classical ergodic theorems guarantee that an evolution is ergodic if it is metrically transitive, i.e., if the energy surface is indecomposable into disconnected manifolds (see e.g., ref. [16]). It might seem hopeless to look for

* Here again, FORTRAN IV is less compact but easier to read: T = 0; IF (Q.EQ.2) Y = 1; IF (CPAST.EQ.1) Y = 1 – Y.

the ergodic properties of this cellular automaton, since we do not know the form of the total energy, let alone the topology of the energy surface in the set of all the configurations. It is not so. The rule Q2R illustrates beautifully the following general point. There are two different reasons for the decomposability of the energy surface. The first is when there exist additional "important exact dynamical symmetries" besides energy (Poincaré's uniform integrals). In Q2R, parity is such a symmetry: a dyssymmetric shape never reaches its mirror-image, despite the obvious degeneracy in energy. A second, and more interesting cause of decomposability of the energy surface is the existence of quantities that become exact constants of the motion for an infinite number of degrees of freedom only. These are the *order parameters*, the classical example of which is the magnetization. It is only in the thermodynamic limit that the magnetization of a ferromagnet at $0 < T < T_c$ is guaranteed not to change its sign because of a fluctuation. A ferromagnet (described, say, by the Ising model) has one order parameter and two degenerate ground states. Spin-glasses [46, 52], on the other hand, have in the thermodynamic limit an infinite number of degenerate ground states, and an infinite number of order parameters are needed to characterize them. In other words, expressed by Hyman Hartman at this workshop, a ferromagnet is to one bit of memory what a spin-glass is to an infinite amount of information storage and complexity.

Let us come back now to the Q2R cellular automaton and consider the set of initial conditions that, for both initial times, are made of rectangles full of ones in a background of zeroes. There are the two major cases depending on the size $a \times b$ of the rectangles.

(A) $a = b$ or a and b are multiple of 4. The cellular automaton starts to lose most of its ones but it soon regains them: the motion is periodic with a short-period (in time steps, about twice the number of cells in the larger side). But as soon as the regularity of the initial configuration is perturbed by one cell, the period becomes extremely long, a manifestation of a long Poincaré recurrence time. The regular initial conditions correspond to the exceptional initial conditions for which the motion is manifestly not ergodic (viz., a non-interacting gas with all the initial velocities parallel to the wall of a cubic box).

It is interesting to observe how the cellular automaton encodes the location of the perturbation (and its size, in case several cells are involved). In many instances the region of the original perturbation loses after a few steps all traces of its special meaning at $t = 0$. The cellular-automaton dynamics seems to have forgotten the site of the perturbation, but in fact it remembers it via a totally delocalized encoding.

(B) $a \neq b$ and a and b are not both multiple of 4. In that more general case, one observes again the rapid drop of the number of ones and soon the system stabilizes and oscillates for a larger number of time steps around a fixed shape that looks, say, like the letter H. When it has apparently exhausted all the allowed configurations around that shape, it rapidly generates a very large number of ones and briefly reaches about 80% of its original population, then it drops again from what appears to be the top of a potential barrier, and falls into an oscillatory motion about a new shape, say the letter X. (A second cycle of identical oscillations around the same shape is forbidden: it would have a merger at its entry-point, also it would mean forgetting the original rectangle. A reversible evolution cannot enter nor leave a cycle.) When the size of the original rectangle increases, the number of different patterns that host the small oscillations in a global Poincaré period increases, as does the time spent around the individual patterns. In the limit of extremely large rectangles, the cellular automaton will remain in the first visited pattern and not wander through the set of allowed configurations. The energy surface of Q2R contains a very large number of deep and narrow potential wells, separated by high barriers. We have thus recovered an important behaviour of spin-glasses. The ingredients of this behaviour are known to be disorder and frustration. The same ingredients can be recognized in Q2R. First, disorder is definitely there. Excluding the trivial fourfold symmetry inherited from the rectangular initial conditions, the system does not exhibit any order when it is at the bottom of a potential well. It displays what seem to be random sequences of scattered ones. Yet, the patterns can be described with a very small amount of information as the result of n steps of evolution following Q2R of a rectangle of size $a \times b$. This apparent disorder has in fact a low algorithmic entropy. This cellular automaton appears to be a good laboratory for the ideas of Bennett and Chaitin [8, 13, 22]. The analogy with frustration, on the other hand, is not so convincing. It might correspond to the fact that the system is reversible but did not start from an eigenstate. It will never reach a stable configuration (as, e.g., "Life" does), but rather will try vainly and forever to satisfy all its bounds. Hyman Hartman brought to my attention the fact that although Q2R behaves *like* spin-glasses (i.e., according to approach (ii)), in general cellular automata cannot faithfully model all the degrees of freedom of these materials (i.e., following approach (iii)). The reason is the disorder in a spin-glass is *quenched* and cannot be represented by a *uniform* cellular-automaton rule. It turns out, however, that non-uniform transition rules can readily be implemented by CAM [56]. This brings us back to approach (i)!

5.2. *Fractals and relaxation to chaos through period doublings*

Wolfram [60] discovered that plots of one-dimensional cellular-automaton evolutions as a function of time often generate beautiful fractal [38] figures. Even when no plot against time is involved, many two-dimensional cellular automata also form scale-invariant patterns. In a generalization of "Life" recently devised by Wainwright, some simple configurations generate fractal flotillas of "gliders." In counting rules, the generic form of fractal patterns is always that of multiple "Maltese crosses" (a cross on each of the four arms of which grow three crosses on each of the four arms of which grow three crosses . . .). These beautiful figures have been observed in 1965 by Fredkin [21] on the "mod 2" rule Q13 and in 1969 by Ulam [7] with the double-counting rule Q1Q01234 applied on a single seed.

The fractals produced by the rectangle-confined rule V2R present an additional physical interest. Starting with a non-convex pattern of ones at $t = 0$ and $t = 1$, the evolution consists in a filling of the concavities by Maltese crosses. This evolution is first periodic both in space and time. But soon the cycles become more and more complicated, the doubling of their temporal period occurs, and eventually the patterns become chaotic both in space and time. Notice that the chaos is not set here by varying a parameter, like in the logistic mapping [10, 29]. The phenomenon displayed by rule V2R is a relaxation rather than a transition to chaos. The double-counting rule V1VR also produces very long sequences of period doublings of oscillating fractal patterns with initial conditions made of several seeds forming a sublattice (a construction readily obtained with periodic boundary conditions).

5.3. *Local conservation laws, Faraday shields, phase velocity, causality and light-cone cooperative behaviour, and non-separability*

There exist patterns (e.g., adjacent ones and zeroes in two consecutive configurations) that are invariant under the evolution of some time-reversal invariant rules. Such patterns cannot appear nor disappear during an evolution, they are characteristic of the initial conditions. Margolus, who discovered these effects on rules V05R and V1234R, calls them *local conservation laws* [39] since they are related to fixed sites. When such patterns form a closed "curve," they partition the array into an interior and an exterior region that do not influence each other: a perturbation outside of a region cannot be perceived inside of it: the invariant curves act like Faraday shields to information propagation. Furthermore, when the curves form a regular sublattice (see above), a regime is soon reached where they emit wave-trains of zeroes and ones the nodes of which travel at a speed larger than a cell per time-step (the "speed of light"). Such "phase velocities" reveal the undamped correlations that occur over the whole array: distant cells do not communicate, yet they seem to "know" about each other's state and conspire to produce large stable coherent patterns. Despite this apparent supraluminal exchange of information, causality is of course not violated. Indeed, these distant cells all belong to the future light-cone of some past event, that constitutes their common *cause*. It is the exact reversibility of the discrete dynamics that forbids all forms of damping and enables the array to feel the full strength of the consequences of this event from the remote past. These undamped correlations are a general feature of reversible cellular automata: the information carried by correlations is always fully conserved, even though conspicuous few-cell correlations generally diffuse, in an apparent irreversible way, into many-cell correlations (cf. section 2.2). These very stable coherent patterns* are reminiscent of the structures produced out of equilibrium by dissipative systems that are driven by external forces (e.g., in the Rayleigh–Bénard's convection [53]). The locally conserved shapes seem to act like sources and sinks of energy, despite the fact our cellular-automaton dynamics is reversible and the array is isolated. This conflict challenges the characterization [44] of a discrete dynamical system as "Hamiltonian" by its obeying Liouville's theorem; or, alternately, it challenges the usual association of large-scale coherent structures and cooperative behaviour with dissipative phenomena.

These correlations that lay dormant for a long time and manifest themselves in the most surprising ways can naturally occur whenever the initial conditions are not set at random. If spatial correlations are present at $t = 0$, the determinism of the dynamics together with the absence of damping carry on the *nonseparability* of the system for an arbitrary length of time. This nonseparability permits in principle a cellular automaton to circumvent (in a rather trivial way, to be sure) Bell's theorem [2, 9] and to exhibit certain characteristics expected only of quantum systems**. (See Feynman [18] for a detailed discussion of cellular automata and Bell's inequalities.)

*See color plates in Taffoli [56], this volume.

**Peres and Zurek [48] have clarified the connection between nonseparability and determinism.

5.4. *More on rule* T13: *a global conservation law, diffusion of the least significant bit, and the hyperbolic behaviour of cellular automata*

Let us write the heat equation in one dimension in the form

$$\frac{\partial}{\partial t} Cu = \frac{\partial}{\partial x}\left(K \frac{\partial u}{\partial x}\right), \tag{5.1}$$

where u is the temperature field, C the specific heat and K the thermal conductivity. A double integration of (5.1) over both space and time is reduced by Green's formula to a *single contour integral*

$$\oint\left(Cu\,\mathrm{d}x + K\frac{\partial u}{\partial x}\,\mathrm{d}t\right) = 0\,. \tag{5.2}$$

This relation expresses the conservation of heat: the difference of heat contained in an arbitrary region between two times equals the sum of the losses across the boundaries between these times. Let us find out now the relevance of this to cellular automata. The simplest discretization of (5.1) reads, i being the spatial index,

$$u_i^{t+1} - u_i^t = u_{i+1}^t - 2u_i^t + u_{i-1}^t\,, \tag{5.3}$$

where we took $C = K$ and the discretization steps to unity. In modulo 2 integer arithmetic, this is nothing but the "mod 2" cellular-automaton rule T13! Similarly, a central difference discretization of the time-derivative in (5.1) would give the second order rule D1R. These rules can then be said to describe the diffusion of the least significant bit. (It would be interesting to observe in this manner the turbulence of the least significant digit in the resolution of Navier–Stokes equation in modular arithmetic.) The derivation of (5.2) from (5.1) applies here too, because the heat equation is linear – and in the linear domain all the results of differential and integral calculus have their exact equivalent in the calculus of finite sums and differences. Summing (5.3) over both space and time yields a cancellation of most terms, an exact discrete counterpart of the elementary derivation of Green's formula. We then get the *single sums*

$$\sum_{i=I_1}^{I_2} (u_i^{T_2} - u_i^{T_1}) = \sum_{t=T_1}^{T_2-1} (u_{I_2+1}^t - u_{I_2}^t - u_{I_1}^t + u_{I_1-1}^t)\,.$$

In modulo 2 arithmetic, this relation equates for T13 the parity difference between times T_1 and T_2 of an arbitrary segment of the array to the parity losses through the boundaries $i = I_1$ and $i = I_2$. This result extends of course immediately to "mod 2" rules in higher dimensions. The simple exact correspondence between heat and parity of bits permits a precise discussion, though in a very restricted context, of the question of conservation and flux of information.

We notice here an important limitation of the simulating ability of cellular automata. We started with a parabolic PDE, but obtained a behaviour typical of hyperbolic PDE: the heat equation transmits information with infinite speed (a rather unphysical feature), but cellular automata produce a heat wave-front that propagates with finite velocity*.

Maybe this limitation could in fact "give a point" to the "fundamentalist" approach (iii), exposed just below? Cellular automata are here in definitive conflict with the continuum modeling, but indeed parabolic equations are as a rule approximate or result from a statistical or macroscopic modeling.

6. Cellular automata as original models of physics

Until now we have taken the expression "simulating physics" in its weaker acceptations, i.e., *computer treatment* (section 3), or *conceptual analogies* (section 5). But "simulating" may be given a stronger meaning, as in the sentence "RCL electrical circuits can *simulate* mechanical oscillations". It is this sense of exact correspondence that Feynman uses in his article "Simulating Physics with Computers" [18]. Trying to use cellular automata to simulate physics in this sense is a bold approach but is in line with the atomistic tradition that advocates taking discreteness seriously (cf. Toffoli's "Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics" [57]). This old tradition has taken on a modern specialization that, stated grossly, sees nature as locally – and digitally – computing its immediate future. Fredkin has exposed this view** at this workshop; see also the articles by Feynman, Finkelstein, Minsky, Wheeler, and Zuse in the proceedings of the 1981 conference on *Physics of Computation, Part II: Computational models of physics* [35].

A major merit of this approach is that it carries to its logical consequences the seemingly innocuous statement that "digital computers can simulate everything". This view is tacitly endorsed by most physicists in their daily intercourse with the computer, yet it conflicts with the standard formulation of physics. Almost twenty years ago, Feynmann has expressed his concern with this conflict.

* To be sure, the same is true with any discrete system, e.g., a digital computer, but the finiteness of the speed of propagation is particularly conspicuous with cellular automata endowed with a small number of states per cell.

** Whereas this view has not affected the current understanding of physics, it has motivated original advances in the theory of computation, viz., reversible logic, the billiard-ball model of computation [19], and a quantum theory of the Turing Machine [3]. These results establish direct mappings between physical (non-dissipative) systems and general-purpose information-processing systems. These mappings (that had for long been "proven" not to exist, [4]) are important for the efficiency-minded computer designer, since they automatically include physical constraints and tradeoffs into a computation model. Such mappings are also of great conceptual interest, since they permit an accurate transfer of notions and methods from physics to the theory of computation [19, 39].

"It always bothers me that, according to the laws as we understand them today, it takes a computing machine an infinite number of logical operations to figure out what goes on in no matter how tiny a region of space, and no matter how tiny a region of time. How can all that be going on in that tiny space? Why should it take an infinite amount of logic to figure out what a tiny piece of space/time is going to do? So I have often made the hypothesis that ultimately physics will not require a mathematical statement, that in the end the machinery will be revealed, and the laws will turn out to be simple, like the chequer board with all its apparent complexities."

An awesome thought! But since we have not so far come across a cellular automaton that can pretend to be an original model of physical phenomena, it is only appropriate to go on with the quote:

"But this speculation is of the same nature those other people make – 'I like it', 'I don't like it', – and it is no good to be prejudiced about these things."

7. Conclusion

Cellular automata are definitely capable of various levels of simulation of physics. Approach (i), defined in the introduction and illustrated in section 3, places cellular-automaton hardware resources [56] between special-purpose machines [36, 47] and general-purpose computers. Because of that particular position in the tradeoff between efficiency (that characterizes the former), and versatility (embodied by the latter), cellular–automaton machines can be seen as general-purpose machines for discrete problems with local interactions. The validity of the simulations by cellular automata of space-discrete systems (lattices) is however conditional to the taming of the effects of the two other discretenesses (time and variables, see section 4).

In contrast with standard simulations, cellular automata do not only seek a mere numerical agreement with a physical system, but they attempt to match the simulated system's own structure, its topology, its symmetries, in short its "deep" properties. The exact computability of cellular automata is a precious asset for the study of these properties. For example, there are fundamental connections between microscopic laws and global behaviour that derive from a system's very reversibility. (Conservation laws, for example, owe their existence to the need to keep the memory of the past.) To investigate these connections it is important to have models whose evolution can be computed *exactly* for an arbitrary length of time. In contrast, if a computation throws away some information about the current state of the system (say, by approximating a real variable by its nearest computer-word representation), it fails to capture the very information-preserving nature of the reversible process one is interested in.

Furthermore, the originality of cellular automata as exactly computable models yields novel insights into old problems. The ergodic theorem, for example, finds its traditional motivation in the fact that it equates a quantity deemed impossible to approximate (average over time) to a quantity calculable in principle with the prescriptions of statistical mechanics (average over the energy surface). For cellular automata the order of difficulty is precisely reversed (section 5.1): there is a priori no access to the geometry of the energy surface; on the other hand, the exact measurement of observables is immediate and the accuracy of their time-average is only a matter of statistics. In the same vein, reversible cellular automata show how non-separability can be very simply related to damping, causality, and the exact conservation of information (section 5.3).

The distinction between approaches (i) and (ii) make many simulations by cellular automata a guaranteed winner. Either the observables of a cellular automaton agree with those of a physical system or model, and approach (i) justifies the uses of that particular cellular automaton; or, as we saw instances in section 5, a cellular automaton has no structural resemblance with any physical system but nevertheless exhibits features that are unquestionably physical. Then approach (ii) demonstrates the power and generality of concepts which, though conceived for a physical problem, cannot only be applied but also defined in contexts totally unrelated to physics. Entropy is of course the classical example of such a concept, and the "success story" of its successive extensions from a strictly thermodynamical quantity to an arch notion in the foundation of statistics [36] seems impossible to match. Yet it appears that with phase transition, order parameter, conservation and loss, physicists have invented notions that turn out to be more powerful than originally suspected. Cellular automata indicate that these notions have a broader scope because they are ultimately based on primitives of computational and informational nature – such as counting, labelling, and comparing – rather than on specialized aspects of physics proper. The physicist who would claim that these extensions of meaning are of little use for his trade might prove to be less guilty than Oswald and Mach in their rejection of Boltzmann's entropy. The computer scientist, on the other hand, cannot legitimately ignore the central role that physics plays in the theory of computation. But this is another story [3, 19, 39].

Most of the physical features of cellular automata presented in this article were obtained by a systematic study of a restricted class – the counting rules. It is urgent to investigate wider classes (double-counting rules, larger neighborhoods, starting maybe with self-conjugate rules) in order to reveal more of these features. But limiting the research to this kind of "physics a posteriori" would be a dubious methodology. In a large measure, cellular automata have been so far a kind of Rorschach test for physicists, who recognize in them their pet models. It is equally urgent to continue the general investigations on discrete modeling. It would be useful, for example, to put the mechanical analogies of section 2.2 on a sounder ground, say, by constructing a discrete Legendre transform that would justify the identification of two consecutive configurations of a cellular automaton based on (2.1) as a point in a phase-space. Also, it would be interesting to know whether the characterization of a discrete dynamical system as Hamiltonian on the basis of its obeying Liouville's theorem is unique, or even valid (see section 5.3). Would a criterion based on a discrete analogue of the symplectic structure define a different kind of discrete Hamiltonian systems? Likewise, it is challenging to generalize the derivation of global conservation laws (section 5.4) to nonlinear rules.

All these questions pertain to the same problem. In recent years, many efforts have been devoted in order to identify and to the study the phenomena that do not depend on the details of the underlying microscopic interactions, but rather on symmetries or on universality classes. It is natural to go on the next step and to try to find out what are the processes, the laws, and the formal structures of physics for which the continuous nature of space, time and of dynamical variables is not essential.

Acknowledgements

This work has benefited from intense daily discussions with my colleagues in the Information Mechanics Group at MIT, E. Fredkin, N. Margolus, and T. Toffoli. The introduction and the conclusion have been adapted in part from grant proposals written jointly by all of us. Most of the research for this article has been made possible thanks to the availability of the Cellular Automaton Machine (the work of Tom Toffoli) and that of the user-friendly software that drives it (the work of Norm Margolus). I am very thankful to H. Orland for his continuous assistance all through this work, for his encouragements and also for his words of caution! I am indebted to H. Hartman for convincing me of the importance, for both physics and the theory of distributed computation, of simulating spin-glasses with cellular automata. I enjoyed useful discussions with C. Bennett, H.

Herrmann, Y. Pomeau, S. Redner, and D. Stauffer. I am also grateful to O. Martin and W. Zurek for helpful correspondence.

References

[1] R. Balian, Du microscopique au macroscopique (Ellipses, Paris, 1982).
[2] J.S. Bell, Rev. Mod. Phys. 38 (1966) 447.
[3] P. Benioff, Int. J. Theor. Phys. 21 (1982) 177; J. Stat. Phys. 29 (1982) 515; Phys. Rev. Lett. 48 (1982) 1581.
[4] J. Bernstein, "Calculators: Self-Replication", The New York Times, February 15, 1976, reprinted in Experiencing Science (E.P. Dutton, New York, 1980).
[5] K. Binder, ed., Monte Carlo Methods in Statistical Physics (Springer, New York, 1979).
[6] K. Binder and D. Stauffer, Adv. in Phys. 25 (1976) 343.
[7] A.W. Burks, ed., Essays on Cellular Automata (University of Illinois Press, Urbana, 1970).
[8] G. Chaitin, "Algorithmic information theory", in: Encyclopedia of Statistical Sciences, Vol. 1 (Wiley, New York, 1982) pp. 38–41.
[9] J.F. Clauser and A. Shimony, Rep. Prog. Phys. 41 (1978) 1881.
[10] P. Collet and J.-P. Eckmann, Iterative Maps on the Interval as Dynamical Systems (Birkhäuser, Boston, 1980).
[11] M. Creutz, L. Jacobs and C. Rebbi, Phys. Rev. Lett. 42 (1979) 1390; Phys. Rev. D20 (1979) 1915.
[12] P.C.W. Davies, The Physics of Time Asymmetry (Univ. of California Press, Berkeley, 1977).
[13] M. Davis, "What is a computation?", in: Mathematics Today: Twelve Informal Essays, L. A. Steen, ed. (Springer, New York, 1978) pp. 241–267. Published in softcover (Vintage Books, New York, 1980).
[14] C. Domb, E. Stoll and T. Schneider, Contemp. Phys. 21 (1980) 577.
[15] F. Dyson, Disturbing the Universe (Harper and Row, New York, 1979).
[16] L.E. Farquhar, Ergodic Theory in Statistical Mechanics (Interscience, London, 1964).
[17] R.P. Feynman, The Character of Physical Law (MIT Press, Cambridge MA, 1967) pp. 57–58.
[18] R.P. Feynman, "Simulating Physics with Computers", Int. J. Theor. Phys. 21 (1982) 467.
[19] E. Fredkin and T. Toffoli, Int. J. Theor. Phys. 21 (1982) 219.
[20] F. Fucito, E. Marinari, G. Parisi and C. Rebbi, Nucl. Phys. B180 (1981) 369.
[21] M. Gardner, Scientific American 223 : 4 (1970) 120; 224 : 2 (1971) 112; 224 : 3 (1971) 106.
[22] M. Gardner, "The random number omega bids fair to hold the mysteries of the universe", Scientific American 241 : 5 (1979) pp. 20–30.
[23] E. Goles Chacc, "Comportement oscillatoire d'une famille d'automates cellulaires non uniformes", Thèse, Grenoble, 1980.
[24] J.M. Greenberg, C. Green and S.P. Hastings, SIAM J. Alg Disc. Meth. 1 (1980) 34.
[25] B. Halperin and P.C. Hohenberg, Rev. Mod. Phys. 49 (1977) 435.
[26] F. Harbus and H.E. Stanley, Phys. Rev. B8 (1973) 1141, 1156.
[27] F. Harbus, A. Hankey, H.E. Stanley and T.S. Chang, Phys. Rev. B8 (1973) 2273.
[28] W.D. Hillis, Int. J. Phys. 21 (1982) 255.
[29] D.R. Hofstadter, "Strange attractors: mathematical patterns delicately poised between order and chaos", Scientific American 245:5 (1981) 22.
[30] A. Hoogland, J. Spaa, B. Selman and A. Compagner, J. Comp. Phys. 51 (1983) 250.
[31] S. Kaufmann, Physica 10D (1984) 145 (these proceedings).
[32] S. Kirkpatrick, Phys. Rev. B116 (1977) 4630.
[33] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, "Optimization by simulated annealing", Science, 220 (1983) 671.
[34] L.D. Landau and E.M. Lifschitz, Mechanics (3rd ed.) (Pergamon Press, Oxford, 1976), section 6.
[35] R. Landauer and T. Toffoli, eds., Int. J. Theor. Phys. 21:6–7 (1982).
[36] R.D. Levine and M. Tribus, eds., The Maximum Entropy Formalism (MIT Press, Cambridge MA, 1979).
[37] W.S. McCulloch and W. Pitts, Bull. Math. Biophys. 5 (1943) 115.
[38] B. Mandelbrot, The Fractal Geometry of Nature (Freeman, New York, 1982).
[39] N. Margolus, "Physics-like Models of Computation," Physica 10D (1984) 81 (these proceedings).
[40] J.E. Mayer and M.G. Mayer, Statistical Mechanics (2nd ed.) (Wiley, New York, 1977).
[41] C. Mead and L. Conway, "Physics of Computational Systems," Chapter 9 of Introduction to VLSI Systems (Addison-Wesley, Reading, Mass., 1980).
[42] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, J. Chem. Phys. 21 (1953) 1087.
[43] T. Niemeyer and J.M.J. van Leeuwen, Physica 71 (1974) 17.
[44] E. Ott, Rev. Mod. Phys. 53 (1981) 655.
[45] G. Parisi, Nucl. Phys., B180 (1981) 376.
[46] G. Parisi, Phys. Rev. Lett. 43 (179) 1754; J. Phys. A 13 (1980) 1101, 1885; Phil. Mag. B41 (1980) 677.
[47] R.B. Pearson, J. Richardson and D. Toussaint, J. Comp. Phys. 51 (1983) 241.
[48] A. Peres and W.H. Zurek, Am. J. Phys. 50 (1982) 807.
[49] J. Hardy, O. de Pazzis and Y. Pomeau, Phys. Rev. A13 (1976) 1949.
[50] P.E. Seiden and H. Gerola, Fundamentals of Cosmic Physics 7 (1982) 241.
[51] R. Shingai, Inf. and Control 41 (1979) 324.
[52] H. Sompolinsky, Phys. Rev. Lett. 47 (1981) 935.
[53] H.L. Swinney and H.P. Gollub, Physics Today 31:8 (1978) 41.
[54] M. Tchuente, "Contribution à l'étude des méthodes de calcul pour des systèmes de type coopératif", Thése, Grenoble, 1982.
[55] T. Toffoli, "Cellular Automata Mechanics", Tech. Rep.

No. 208, Logic of Computers Group, CCS Dept., The University of Michigan (November 1977).

[56] T. Toffoli, "CAM: A High-Performance Cellular Automaton-Machine", Physica 10D (1984) 195 (these proceedings).

[57] T. Toffoli, "Cellular Automata as an Alternative to (rather than an Approximation of) Differential equations in modelling Physics", Physica 10D (1984) 117 (these proceedings).

[58] G. Toulouse, Commun. in Physics 2 (1977) 115.

[59] N.G. van Kampen, Stochastic Processes in Physics and Chemistry (North-Holland, Amsterdam, 1981).

[60] S. Wolfram, Rev. Mod. Phys. 55 (1983) 601 and Physica 10D (1984) 1 (these proceedings).

102655669

Physica 10D (1984) 117–127
North-Holland, Amsterdam

CELLULAR AUTOMATA AS AN ALTERNATIVE TO (RATHER THAN AN APPROXIMATION OF) DIFFERENTIAL EQUATIONS IN MODELING PHYSICS*

Tommaso TOFFOLI
MIT Laboratory for Computer Science, 545 Technology Sq., Cambridge, MA 02139, USA

Cellular automata are models of distributed dynamical systems whose structure is particularly well suited to ultrafast, exact numerical simulation. On the other hand, they constitute a radical departure from the traditional partial-differential-equation approach to distributed dynamics. Here we dicuss the problem of encoding the state-variables and evolution laws of a physical system into this new setting, and of giving suitable correspondence rules for interpreting the model's behavior.

1. Introduction

1.1. *Preview*

We shall present a train of thoughts that in summary runs more or less as follows. (a) There are novel computational resources which on certain tasks may outperform conventional resources by very, very many orders of magnitude. (b) In comparing the two classes of resources, it becomes obvious that the conceptual development of mathematical physics must have been strongly influenced by the nature of the available computational tools. (c) The new resources suggest a new approach to the modeling and simulation of physical systems; in particular, it is possible to replace the customary concepts of real variables, continuity, etc., with more constructive and "physically-minded" counterparts.

1.2 *Infinities in mathematical physics*

Mathematical physics, both classical and quantum-mechanical, is pervaded by the notion of a "continuum," that is, the set R of real numbers with its natural ordering and topology. Maxwell's equations provide a typical example. There, space is a structure S diffeomorphic to R^3, and the electromagnetic field at each point is an element of $Q = R^6$, so that the phase space for the whole field is $Q^S = (R^6)^{R^3}$, a very uncountable state set! On this phase space, we assign a dynamics in the form of a group of transformations T (time) indexed by R.

How do we manage to specify in some constructive way the behavior of a system beset by so many uncountable infinities? Part of the answer is that we do not deal with the "generic" system. Rather, we concentrate on systems having such very special properties (e.g., continuity, uniformity, locality, linearity, or reversibility – Maxwell's equations happen to have all of these properties at once†) that most of the infinities "cancel out," so to speak, and we can make some definite qualitative or quantitative statements about the system's behavior. Of those special properties, the most important for taming infinities is certainly *continuity*. Intuitively, a small uncertainty about the system's initial state leads to a correspondingly small uncertainty about its final state, so that we don't have to worry about capturing its state with "infinite" precision, whatever that may mean. More precisely, in mathematical physics, even when we choose for technical reasons to represent states as encoding an infinite amount of *information*, the

* This research was supported in part by the Defense Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contracts Nos. N00014-75-C-0661 and N00014-83-K-0125, and in part by NSF Grant No. 8214312-IST.

† In particular, the elements of T are continuous with respect to the topology of Q and commute with the elements of S.

temporal *correlations* between states introduced by the dynamics may be much more finite*.

In conclusion, in modeling physics with the traditional approach, we start for historical reasons (see below) with mathematical machinery that probably has much more than we need, and we have to spend much effort disabling or reinterpreting these "advanced features" so that we can get our job done *in spite* of them. On the other hand, in this paper, we outline an approach where the theoretical mathematical apparatus in which we "write" our models is essentially isomorphic with the concrete computational apparatus in which we "run" them. Starting from this finitary approach, the few infinities that we may still want to incorporate in a physical theory (e.g., as the size of a system grows without bounds) are defined as usual by means of the limit concept. However, in our approach the natural topology in which to take this limit is that of the *Cantor set*, rather than that of the real numbers.

1.3. *Old and new resources*

Traditional computation, whether by man or machine, involves the sequential processing of a few dozen or at most a few thousand "objects." In symbolic computation the objects are formulas and the processing is done by means of derivation rules, while in numerical computation the objects are finite numbers and the processing entails algebraic operations. (At a more microscopic level both kinds of computation use set operations on very small sets of symbols; however, both computers and people come already hardwired to perform higher-level operations on larger data "chunks.")

Let's consider numerical computation as performed by ordinary computers. As long as fast computer memory is very expensive (as was the case until recently), it is necessary to encode information about a system's state in a very compact way. If an n-bit machine word can encode 2^n different states, then, for instance, we use each one of these states to represent a different value for a scalar quantity†; thus we arrive at the INTEGER or REAL variables, of, say, FORTRAN, where a portion of the real line is chopped up into a number of segments‡ and a different binary code is assigned to each segment. Representation compactness is bought at a price; i.e., processing of these variables requires a rather complex piece of machinery called an "arithmetic/logic unit" (for INTEGER variable), or a much more complex piece of machinery called a "floating-point" processor (for REAL ones); the latter mechanism can be simulated by a lengthy program running on the arithmetic/logic unit. The cost of such hardware is many orders of magnitudes larger than that of a memory word, and thus the customary approach is to time-share it among the few thousand (or million) words that make up the memory.

We shall consider now a different approach. Today, *pure* memory, i.e., without input/output buffering and access circuitry (such as the customary binary-addressing tree), is essentially a free commodity: at one bit per micron square, one could pack ≈ 1. Giga ($\approx 2^{30}$) bits on a 1-inch chip. Then let's be bold, and decide to use some sort of *unary* – rather than *binary* – notation to encode a scalar variable. That is, the value of the variable will be just the number of ones in a certain portion of memory; that's extremely lavish – an integer that used to occupy a 16-bit word will now take 2^{16} bits! However, this extravagance in *storage* buys us certain advantages in *processing*. A scalar variable is now just a "bag" of ones (the position of each bit is irrelevant: each bit has the same weight), and to add two bags we can just "pour" their contents together. As we shall explain in detail later, variables that are encoded in a distributed, local, and

* Cf. a very clear discussion by Everett [2]. The salient point is that "the amount of information in a state" is not as important a concept as "the amount of correlation between two states." While information is measured in a way that has a certain amount of arbitrariness (it depends on the "gauge" chosen), correlation is a "gauge-invariant," absolute quantity.

† We call *scalar* a quantity whose values are naturally ordered and spaced on a linear scale – as contrasted to quantities that range over an unstructured set.

‡ These segments are of uniform width for INTEGER variables, and exponentially increasing width for REAL ones.

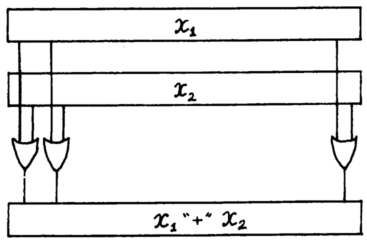


Fig. 1. The "sum" of two unary-encoded scalar variables, as performed by bit-wise ORing.

uniform way (as our bags of ones) naturally lend themselves to processing that is distributed, local, and uniform.

For instance (this is a very naive example – don't laugh but read on!), suppose we have two of these 2^{16}-bit words that we want to add. Let's be extravagant again, and attach a miniature arithmetic/logic unit to each bit. This will be a simple OR gate, no more complex than the FLIP-FLOP which realizes a memory cell. Then we can proceed as in fig. 1.1. After just one propagation delay, the result register will contain the "sum" of the two words. Note that, in this "sum," 1's that are in a homologous position in the two words will contribute only 1 – rather than 2 – to the total. Though the relative error decreases very fast as the 1's in the two words become sparse (intuitively, when our universe consists mostly of vacuum), this is still an approximate and rather inefficient way of doing things, and that's why we call the example "naive." The important point is that all the processing is local. There is no feedback, no carries, no long lines that traverse the whole chip and whose capacitance we have to charge and discharge. All can be done in a fraction of a nanosecond. We wasted many powers of two by using unary rather than binary encoding; but we recoup many powers of two from the fact that in our scheme memory only needs local access and thus can be somewhat denser, and processing is done locally and thus can be extremely faster. It is this lack of overhead that makes our approach attractive for many kinds of physical computation, where one deals with systems that are inherently distributed over spacetime and subjected to laws that are local and uniform.

In the above example we applied local processing to a machine word which encoded a scalar variable and which would typically represent a *lumped* quantity of a physical model. But lumping is usually introduced artificially, in order to adapt a problem to the techniques of ordinary numerical computation. However, if we are able to do local processing at an extremely fine scale, we might as well directly construct our models, much more naturally, as ones in which variables and parameters are *distributed.* We shall introduce a quite general class of such models and discuss their relevance – in the light of the new computational resources – for both theoretical and practical mathematical physics.

2. Cellular automata vs. differential equations

2.1. *Generalities*

We assume some familiarity with the concept of "cellular automaton"*.

In the *cellular-automaton* model of a dynamical system, the "universe" is a uniform checkerboard, with each square or *cell* containing a few bits of data; time advances in discrete *steps*; and the "laws of the universe" are just a small look-up table, through which at each time step each cell determines its new state from that of its neighbors; this leads to laws that are *local* and *uniform.* Such a simple underlying mechanism is sufficient to support a whole hierarchy of structures, phenomena, and properties. Cellular automata provide eminently usable models for many investigations in physics, chemistry, and biology, as well as for experiments in combinatories and for studies in parallel computation [8].

Many theoretical results have appeared on cellular automata. Yet, the fundamental problem (as in the case of partial differential equations, of which

* A well-known example of a cellular automaton is John Conway's game of "life" (cf. Martin Gardner, "The fantastic combinations of John Conway's game 'life'," Scientific American, **223:4** (1970) 120–123). This cellular automaton was shown by Bill Gosper to be computation-universal. For a systematic introduction to cellular automata, refer to Toffoli [8].

cellular automata represent a discrete counterpart) is to determine the temporal evolution of the system, and this problem in general does not have an analytical solution. That is, in order to describe the state q_t that the system will attain in t time-steps starting from a given initial state q_0, the only general method is to construct one-by-one the intermediate states $q_0, \ldots, q_{t-1}$, i.e., to perform numerical integration. Note, however, that because of cellular automata's intrinsic discreteness, here numerical integration is an *exact* process (there are no truncation or round-off errors to worry about), and the results that one obtains have thus the force of *theorems*. In other words, any properties that one discovers through simulation are guaranteed to be properties of the model itself rather than a simulation artifact.

2.2. *Physical realization of cellular automata*

Much as Gutenberg developed a way to reproduce in an arbitrary number of copies *static* information-bearing structures such as text, the introduction of integrated-circuit technology has made it possible to replicate at will *dynamic* information-processing structures such as electronic circuits.

There are four main factors that determine the cost/performance ratio of an integrated circuit, namely, circuit design and layout, ease of mask generation, silicon-area utilization, and maximization achievable clock speed; for a given technology, the latter is inversely proportional to the maximum length of critical signal paths. In terms of these four parameters, cellular automata are perhaps the computational structures best suited for a VLSI realization. In fact, circuit design reduces to the design of a single, relatively simple cell, and layout is uniform; the whole mask for a large cellular-automaton array (that is, not only the cells with their internal connections but also the interconnections between cells) can be generated by a step-and-repeat procedure; essentially no silicon area is wasted on long interconnection lines; and, because of the locality of processing, the length of critical paths is minimal and independent of the number of cells.

Ignore, for a moment, what it is that a cellular automaton actually computes, and whether it can be put to any good use. The fact remains that if I put one-million dollars' worth of cellular-automaton VLSI circuits in a black box, and ask somebody to simulate its behavior with one-million dollars' worth of general-purpose computer, their simulation will be perhaps 10^{12} (one million million) times slower. The challenge is, of course, unfair, because a general-purpose computer is optimized to do other things, but that is exactly the point! In other words, with suitably realized cellular automata one can see things that cannot be seen any other way. Whether these things are worth seeing – well, that's another matter, and this paper attempts to make educated guesses about it. Probably the issue can only be judged *a posteriori*.

In this context, we have constructed a special-purpose cellular-automaton machine [9] which, although based on serial processing, is about a thousand times faster that a general-purpose computer programmed for the same task. Experiments in parallel dynamics using this machine have been very rewarding (cf. Vichniac [10]), and we have confirmed at least in a qualitative way the feasibility of the approach discussed in 2.3 below*.

2.3. *Partial differential equations in spacetime*

Let us consider a partial differential equation with space- and time-independent parameters; for concreteness, let us choose the *heat* equation

$$c\frac{\partial T}{\partial t} = k\nabla^2 T. \tag{1.1}$$

This is a mathematical model which is widely used for two reasons: (a) it may represent passably well, at a certain level of description, the behavior of a physical system, and (b) we have a rich catalog of techniques for making *mathematical* deductions from it. To what extent these deductions apply to

*Though from an abstract viewpoint speed is irrelevant, imagine having to do actual genetics research using generations of *elephants* rather than of *Drosophila*!

the *physical* system itself depends on how good correspondence (a) is.

For instance, $T(x, t)$ in (1.1) is a real-valued function defined at each point of (abstract) space and time. In the solution of (1.1), even if we assign T at time $t = 0$ in a quite arbitrary way (for instance, as a sum of step functions), T will be a *continuous* function of x for any $t > 0$. This is important, since operationally we cannot measure a temperature *at a point*; we can only measure the mean temperature over a *finite volume*. The correspondence between system and model is then set up at this level; after that, a point temperature is defined, *within the model*, as the limit of mean temperature as the volume goes to zero. Continuity guarantees that this limit exists.

Now, for volumes that are not too small the correspondence between measured mean temperature and its mathematical counterpart works well, in the sense that as we make the volume smaller the measured values fall within a smaller and smaller interval. However, beyond a certain point this correspondence breaks up, and the smaller we make the volume the wilder are the results that we get. This applies not only to continuity in space but also to continuity in time: a temperature probe will reveal larger and larger fluctuations as its thermal inertia is made smaller.

In conclusion, if eq. (1.1) manages to model well, in the large, certain physical systems, it does so not because it rests on the axiomatics of the calculus, which are not shared with the physical system, but because it must somehow capture other essential properties of a diffusion process, such as locality of effects, conservation of certain quantities, etc. Other mathematical approaches might be as (or more) successful at modeling a physical system in the large, and at the same time provide a better insight into a system's microscopic behavior.

The great advantage of differential equations, such as (1.1), is that we have three centuries' experience with methods for their symbolic integration. As long as all computation had to be done by hand, it paid to stylize the physics in a certain direction so as to be able to handle the resulting mathematics. But few differential equations have a closed-form solution anyway, and the past fifty years have seen numerical computation make bolder and bolder claims at being recognized as an essential part of mathematics.

The moment one gives up symbolic manipulation as a major motive for using differential equations, one starts wondering whether one should still keep them as the starting point for numerical modeling. In fact, they lead to concrete numerical computation (e.g., as run on a general-purpose computer) that is at least three levels removed from the physical world that they try to represent. That is, first (a) we *stylize* physics into differential equations, then (b) we force these equations into the mold of discrete space and time and truncate the resulting power series, so as to arrive at *finite-difference equations*, and finally, in order to commit the latter to *algorithms*, (c) we project real-valued variables onto finite computer words ("round-off"). At the end of the chain we find the computer – again a physical system; isn't there a less roundabout way to make nature model itself?

2.4. *The origin of scalar quantities*

The present subsection introduces in an informal way the main point of this paper.

A partial differential equation whose independent variables are space and time, such as (1.1) above, is translated into a finite-difference equation by the following process: (a) continuous space and time are replaced by a discrete grid, (b) the system's state at each point remains a continuous variable of the same kind (e.g., real, complex, vector) as in the original equation, and (c) derivatives are replaced by differences between state-variables that are contiguous in space and time*.

When one translates the finite-difference equation into a computer algorithm, all one does is (a) *discretize* also the real variables, and further (b) restrict them to a *finite* range. State variables are then represented by finite, though quite large, sets.

* The choice of suitable differences, ostensibly made according to definite "correspondence rules," actually requires a certain amount of black magic to be really successful; cf. Labudde and Greenspan [3] for an interesting discussion.

The dynamics of the system is expressed by essentially the same difference rules, with the proviso that overflow or underflow will abort the computation; if the state set is made too small this aborting will happen so often as to make the method useless.

In terms of structure, if not of interpretation, a cellular automaton is a computational scheme just like the above. The only difference is that the variables at each point of the grid are only allowed to range over a very small set – say, two states per cell, "0" and "1." This, however, has profound consequences.

Clearly, with only two states per cell, it is out of the question to think of a cell as an approximation of a scalar variable (as we did with cells of type REAL or INTEGER in FORTRAN), or to think of a Boolean expression involving the states of adjacent cells as an approximation of a real function of real variables (as we did with FORTRAN's algebra). Intuitively, if a picture's essential features are on the same scale as the picture's "grain", then we have no picture at all. Now, we could program the cellular automaton (by choosing a suitable local rule and suitable initial conditions) to simulate a conventional difference-equation scheme; certain blocks of cells would represent machine words, other blocks would realize arithmetic/logic units, etc. – but this is very unnatural and extremely inefficient. Instead, we shall try an original – and much more natural – approach.

As an aid to intuition, we shall think of 1's as "balls" floating in a "vacuum" of 0's. Let us consider the *mean density* α_V over a certain volume V, i.e., the fraction of cells – within that volume – that are occupied by a ball. α will always be a number between 0 and 1. If V consists of only one cell, then α can take on only two values, that is, either 0 or 1. If V consists of – say – 100 cells, then the possible values for α will sample the interval [0, 1] much more finely: 0.00, 0.01, . . . , 0.99, 1.00. As the size of V grows toward infinity, the range of α approximates better and better the unit on the real-line.

However, in order to speak of a "density field"* we would like to define the density *at a point*. Let x be a point of the grid, and $V_{x,r}$ the "sphere"† of radius r and center on x. Let $\alpha_{x,r}$ be the mean density in this sphere. For the moment, we shall associate with a point x the whole sequence of mean densities $\alpha_{x,1}, \alpha_{x,2}, \ldots$ (without attempting to take its limit as r goes to infinity).

A few remarks are in order. (1) There is a trade-off between spatial resolution and resolution in the density domain. If r is small, we are looking at a definite place, but density is coarse-valued; to get finely-spaced values on the density scale we have to look at a large volume. (2) Let us take a random configuration (of cell states for the cellular automaton). For a fixed r, let us study how the density $\alpha_{x,r}$ varies as we move in space. If $r = 1$ (one-cell radius), then as we move x we get for $\alpha_{x,r}$ a sequence of 0's or 1's, with no correlation between the elements of the sequence. As r increases, the values of α will move up and down the unit interval in a smoother and smoother fashion, so that in the limit we can speak of a continuous function. This continuity is not imposed from above, but arises quite naturally if we observe that large spheres centered on neighboring points have much overlap, and thus share most terms in the summation that defines mean density. (3) We shall see later that, once we introduce a dynamics, we encounter an analogous discontinuity in the small and continuity in the large as we move in *time* rather than in space. (4) Here, we are using unary notation to represent the scalar quantity α, as in the example 1.3, but without committing ourselves to computer words – or "bags" – of a definite size. Moreover, under these circumstances the unary representation is not as wasteful as it might appear on first sight. When we want high resolution, and thus must use large bags, it turns out that most of the bits that make up one bag are shared by the neighboring bags (cf. (2) above); no matter how

* The concept of "density" in this discussion can be taken as a prototype for other scalar variables such as *pressure*, *temperature*, etc.

† The exact shape of this "sphere" does not matter. On an orthogonal lattice one might as well take a cube.

large r is chosen, encoding is done at a *constant* cost of 1 bit per bag! (5) At least so far as the present static picture is concerned, the properties of "density" as defined in our model parallel quite closely those of any of the so-called "point variables" of actual physics (e.g., charge density, temperature); indeed, the latter are "smooth" statistical constructs based on an actual "granular" substrate, and lose their meaning when one, attempting to take a microscopic limit, undermines their very statistical base. (6) Finally – and this is an essential point – the practical trade-off for using small-valued variables interacting only locally is that with the same bulk amount of computer space and time we can handle a grid that is thousands of times finer both in space and in time (cf. 1.3, 2.2). Thus, even though the interactions between such simple cells cannot but be elementary, we can hope to synthesize quite complex interactions through massive iteration. We know that the Gaussian curve can be handled by the mathematician by means of symbols on the paper* and can be drawn to any approximation by the numerical analyst; but whenever we find this curve in nature we don't see the mathematician or the analyst – we see an unsteady hand shooting at the same target over and over and over... (cf. Borel [1] for a very relevant discussion).

And now let's introduce some dynamics. To be specific, assume that balls interact according to some definite local rule but maintain their identity. For example, Norman Margolus has constructed a very clever "billiard ball" cellular-automaton rule of this kind [6] in which balls are conserved, undergo elastic collisions, and all travel at the same speed in one of four possible directions (except during a collision, where they slow down for a moment before bouncing back); this rule also supports clumps of balls that stick together and act as hard mirrors. Margolus' rule is strictly *reversible* – i.e., any configuration for the whole cellular automaton has exactly one predecessor (as well as exactly one successor as in any *deterministic* rule), and *computationally universal.* Reversibility guarantees, among other things, that the relations between microscopic and macroscopic behavior satisfy the laws of thermodynamics, while universality implies that in general there is no analytical shortcut to the system's dynamics – in other words, that there is no better way to tell what the system will do than let it do it and watch it!

* But already its integral is not expressible in terms of elementary functions.

Well, if we watch very closely a cellular automaton like this we see a binary computer in operation. But let's stand a certain distance away, and we will see clouds in all shades of gray pulsating and swirling and colliding and mixing.... In other words, if we associate with each point in space the "level of gray," (i.e., the mean density) in its vicinity – rather than just the Boolean value of the cell at that point – this scalar point-variable evolves smoothly, much as if it were "driven" by a differential equation.

Let's see what is involved in this new interpretation. For a fixed r, we have a *density field* $\alpha_{x,r}$. If $r = 1$, this is a Boolean-valued field whose evolution is deterministic and given directly by the cellular automaton rule. If $r \gg 1$, we have a scalar field whose evolution is nondeterministic; however, knowing the field at neighboring points, we can reconstruct from the cellular automaton rule the probability distribution $P(\Delta\alpha)$ that the field will change by an amount $\Delta\alpha$ in one time-step. If P is very sharp, we have a mechanism that is substantially identical to a finite-difference algorithm. We may expect P to be sharp when (a) the rule is suitably chosen, (b) the value of r was selected within a suitable range, and (c) the value of the field is not too close to 0 or to 1. (In our interpretation, 0 and 1 correspond to, respectively, "vacuum" and "infinite" density. Near these extremes our scheme fails to model a finite-differential equation because P will not be sharp enough to give an essentially deterministic result; on the other hand, near the same extremes a FORTRAN program will fail because of overflow or underflow conditions.)

In conclusion, we have an efficient computational mechanism based on microscopic *prim-*

itives; in this model one can introduce in a rather natural way, as *derived* constructs, macroscopic scalar quantities which, given an appropriate microscopic dynamics, behave much as the quantities predicated by the differential-equation model. The two models are definitely different, but the area of overlap permits one to establish "correspondence rules" between them, so as to arrive at criteria for determining to what extent a microscopic dynamics is indeed "appropriate" for generating the desired macroscopic behavior.

2.5. *A counting argument*

The variety of differential equations that one can write is enormous. In a cellular automaton, on the other hand, once we select the neighborhood on which the new state of cell will depend, all the choice we have for synthesizing the desired behavior is in assigning entries in the look-up table that defines the local map. With few states per cell, this choice doesn't seem too wide. For example, if the new state of a cell depends on the current state of the cell itself and of its immediate neighbors (say, North, South, East, and West) – five neighbors in all – then, with two states per cell, the table will consist of only $2^5 = 32$ binary entries. No matter how cleverly one assigns these entries, one certainly can't do much with the material at hand.

However, even in this simple case the number of different laws that one can write is 2^{32} ($\approx$ one billion!). With nine neighbors (as in the game of "life"), this number climbs to 2^{2^9} ($\approx 10^{150}$), more than one could explore in the universe's lifetime. Of course, many of these will be trivial variations on the same theme, and most will be utterly uninteresting; but at least we know there is plenty of room to play.

To have even more choice, one can enlarge the neighborhood and use more than two states per cell. However, in the many hours we have spent trying to construct rules that would do what we wanted, we have learned that blind exploration of such an enormous territory is not very rewarding. There are better ways to expand the number of choices in a structured fashion, with more predictable results, making explicit use of analogies from physics or of known combinatorial results. For example, one can make rules that are second-order in time (a class of these automatically yields behavior that is invariant under time reversal); one can make them dependent on the parity of space or time (odd or even steps, or black or white squares on the checkerboard); one can compose into a sequence ("microcode") a small number of different rules involving few neighbors; etc.

After all, even though there exists an uncountable number of differential equations, the set of those that we can explicitly write down is only *countable*, and so is the set of cellular-automaton rules. In conclusion, even though the language of cellular automata uses different primitives than differential equations, there is no a priori reason why it shouldn't have comparable expressive power.

As it happens, we have discovered extremely simple cellular-automaton rules for the "heat" equation (a first-order partial differential equation) and the "wave" equation (second-order). These two equations are the cornerstones of much mathematical physics. (See color plates enclosed in [9].)

3. The concept of continuity in the dynamics of cellular automata

3.1. *Generalities*

The above considerations suggest that, in spite of their discreteness, cellular automata may still support some concepts of continuity and metric, but not the same as in the real-number topology. Then, we shall look elsewhere. We shall start with some miscellaneous considerations.

Observe, first, that while the set of cell-states is *finite* and the set of cells is *countable*, the set of all configurations (i.e., the phase space) is uncountable, and indeed has the cardinality of the real-number *continuum*. Thus, our phase space is as large as that of finite-difference schemes (in spite of the fact that these use real numbers for cell states). In other words, we have as much infinity to play with as the other guys – only ours is organized in a different way, and locally things are always finite.

Second, continuity means, intuitively, that we can choose states that are so close that their successors are still arbitrarily close. But cell states belong to a finite set, so that there is a discrete jump between one and the other. How can one have an arbitrarily small distance between states in this situation?

Third, all cellular automata having the same "format" (grid shape, number of states per cell) have the same phase space. On the other hand, they may have very different dynamics. In each dynamics the trajectories will interconnect the points of phase space in a totally different way. Can we hope to find a single phase-space topology that is natural and relevant to all of these dynamics?

3.2. *The Cantor-set topology*

Consider the generic cellular automaton. Its *cell-state set A* ("*A*" for alphabet") is a finite, unstructured collection of symbols, and cannot but be given the discrete topology. Its *phase space* C ("C" for "configurations") is the Cartesian product A^S of countably many copies of A, indexed by the elements of the *space group S* (i.e., the grid's symmetry group). If S were finite, then C would naturally inherit the discrete topology; but for an infinite index set the natural topology for the Cartesian product is the *Tychonoff product topology*, which is coarser than the discrete topology. In the product topology, open sets can be visualized as follows. Let us assign definite values to a finite number of cells, and consider the set of all configurations that match the given assignment (i.e., the values of all other cells are "don't care's.") Call such a set a *pattern*. Then an *open* is an arbitrary collection of patterns.

If the terms of the countable Cartesian product are finite sets (having, of course, more than one element), as in our case, then the Tychnoff product topology coincides with the topology of the familiar *Cantor set* (. . . take the unit segment on the real line, remove the middle third, and iterate on what is left.) Thus, we get the same "Cantor-set" topology for *all* nontrivial cellular automata (i.e., those having at least one dimension and at least two states per cell).

3.3. *A topological characterization of cellular automata*

Now, one can prove the following. Let τ be the automaton's global map (i.e., its dynamics, or the generator of the time group). Then, for all cellular automata,

Property 1 (*continuity*). τ is *continuous* with respect to the Cantor-set topology [7].

We also know that, by definition,

Property 2 (*commutativity*). τ *commutes* with any element σ of S (the space group)

(Briefly, time and space commute.) Finally, for every cellular automaton

Property 3 (*local finiteness*). There exists a continuous function $q: C \to Q$, where Q is a finite set, such that, if c, c' are distinct configurations, there exist a shift $\sigma \in S$ for which $q\sigma(c) \neq q\sigma(c')$.

By taking $Q = A$, q can be interpreted as a "window" function which projects a configuration on the coordinate axis of a selected cell. This obvious property is used in proposition 1 below to rule out certain pathological cases*.

The important fact in all of this is that, among the dynamical systems consisting of the Cantor set with a dynamics τ and a discrete group of transformations σ,

* I am indebted to Leonid Levin and Douglas Lind for formulating it and suggesting its use in proposition 1.

Proposition 1. Properties 1–3 above constitute a complete characterization of cellular automata.

Thus, we see that, in addition to local finiteness and commutativity between space and time (properties which are put in the very definition of a cellular automaton), *continuity in the Cantor-set topology* is the characterizing property of the dynamics of cellular automata. (The various components of this characterization were rediscovered in bits and pieces by several workers in cellular-automaton theory – including myself [8] – but had been known for a while, under the heading of "shift dynamical systems," to more professional mathematicians [4, 5].)

3.4. *The local point of view*

We shall try to interpret the results of the above subsection.

The main point is that to understand what goes on in a cellular automaton it is not necessary to look at an entire, infinite configuration. Rather, one's attention can be turned to specific place, and one's scope should be widened, in concentric circles, so to speak, only as longer and longer evolution times are considered. Thus, the customary picture which represents the state of a system as a point tracing an orbit in phase space is somewhat misleading: in general, we cannot handle in a finitary way the exact position of the point itself (which encodes an infinite amount of information); on the other hand, we can project the point on a finite subset of axes, and we can enlarge this subset as need requires. We shouldn't try to (and, at any rate, we *can't*) take in the whole picture!

We shall give but one example of the "conspiracy" that forces us to take a local viewpoint. The Cantor set is a *metric* space, that is, it admits of metrics compatible with its topology. What does this repertoire of metrics have to offer? We are faced with the problem of finding a satisfactory metric (a yardstick for "closeness") for a *uniform* system that extends infinitely in space. Because of spatial symmetry, all cells "look the same;" intuitively, we would require of our metric that if we change a 0 into a 1 in a given cell, we should move away a certain distance in phase space, and this distance should be the same no matter which cell we choose. But it turns out [8] that none of such "uniform" metrics is compatible with the Cantor-set topology; in which direction should we relax our requirements?

Here is one way. In spite of being immersed in a uniform sea of cells, we shall pick one arbitrarily. By what criterion? Well, by where *we* are! In comparing two configurations, we make a list of the places where they don't match; a mismatch occurring "here" will be given a weight of $\frac{1}{2}$, and in general any mismatch occurring within a shell of thickness 1 and radius r will be given a weight $2^{-(r+1)}kr^{d-1}$ (where d is the dimensionality of the space), so that successive shells will contribute at most 1/4, 1/8, etc. Thus, the distance between two identical configurations will be 0 and the distance between two configurations that differ everywhere will be 1. This metric is compatible with the Cantor topology. In this metric, two configurations get closer and closer as the nearest point where they differ moves away from us*. Quite seriously, we could characterize the Cantor topology as the topology of self-centeredness. What is nice is that other observers, with their own center of interest different from ours, may choose their own version of the metric, but nonetheless we will all agree on the same topology.

4. Conclusions

We have presented and motivated a new mathematical approach to the modeling of distributed physical systems. This approach is suggested by the availability of new, high-performance simulation

* To make the connections with the traditional δ–ϵ criterion for continuity, recall an obvious property of cellular automata, i.e., that the speed of propagation of information is bounded. If two configurations coincide within a radius r, and thus have a distance less than $\approx 2^{-r}$ then their successors will coincide within a radius of at least $r-1$, and thus their distance will be less than $\approx 2^{-(r-1)}$. This is all that is needed to arrive at a δ–ϵ criterion.

tools, and yields models whose formal structure closely matches that of the available computational resources.

In particular, we have discussed a concept of continuity that is adequate for an operational approach to physics over the whole range from macroscopics to microscopics, and yet does not postulate, like differential equations, an infinite amount of information within a finite volume.

Acknowledgements

I am greatly indebted to Edward Fredkin for giving the start to this train of thoughts, and to two anonymous referees for many helpful suggestions.

References

[1] Émile Borel, Probabilities and Life (Dover Ps, London, 1962).

[2] Hugh Everett, "The Theory of the Universal Wave Function", The Many World Interpretation of Quantum Mechanics, DeWitt and Graham, eds., (Princeton Univ. Press, Princeton, 1973) pp. 3–140.

[3] R.A. Labudde and Donald Greenspan, "Discrete Mechanics—A General Treatment", J. Comput. Physics 15 (1974) 134–167.

[4] G.A. Hedlund, "Endomorphysms and Automorphisms of the Shift Dynamical System", Math. Syst. Theory 3 (1969) 320–375.

[5] H.B. Keynes and J.B. Robertson, "Generators for Topological Entropy and Expansiveness", Math. Syst. Theory 3 (1969) 51–59.

[6] Norman Margolus, "Physics-like Models of computation", Physica 0D (1984) (these proceedings).

[7] D. Richardson, "Tessellations with Local Transformations", J. Comput. System Sci. 6 (1972) 373–388.

[8] Tommaso Toffoli, "Cellular Automata Mechanics", Tech. Rep. No. 208, Logic of Computers Group, CCS Dept., The University of Michigan (November 1977).

[9] Tommaso Toffoli, "CAM: A High-Performance Cellular Automata Machine," Physica 10D (1984) 195 (these proceedings).

[10] Gérard Vichniac, "Simulating Physics with Cellular Automata", Physica 0D (1984) (these proceedings).