#3603

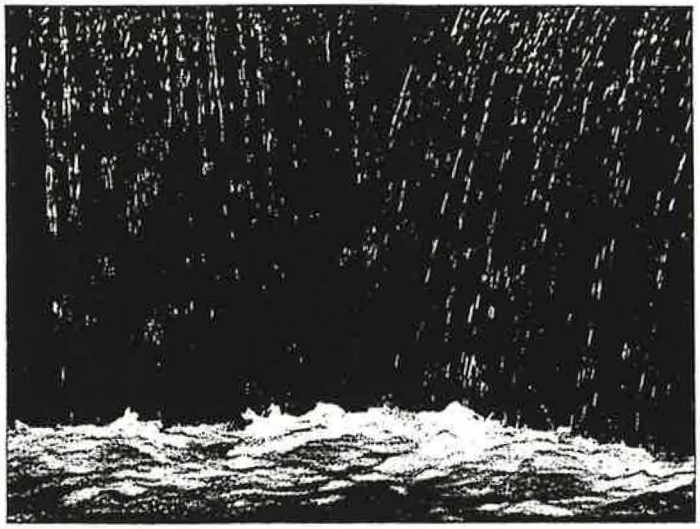
AWC

life and times of cellular automata

Take some simple directions and apply them to a regular pattern of cells on a computer screen. Then watch them play a complex game that mirrors the real world

Greg Wilson

HOW SIMPLE can a model of the real world be and still be useful? During the past decade, physicists and computer scientists have been studying a class of mathematical models known as "cellular automata" which are extremely simple, but can reproduce very complex phenomena.



Tony Stone/Tony Craddock

At normal scales, the real universe is continuous—time and space are smooth, and most physical properties, such as temperature and mass, can take on any value. In a cellular automaton, on the other hand, time is divided into a sequence of discrete ticks, and space into a regular grid of cells. Each of these cells is allowed to take on only one of a small set of values. These are usually represented by a set of small integers, or by different colours on a computer screen.

At each time step, every cell in the cellular automaton updates its state according to rules that depend on its current state and the states of its neighbours. Different grids, different sets of states, or different update rules produce different cellular automata. The simplest are lines of cells that have two states—on and off. One possible update rule for such a one-dimensional cellular automaton is:

$$c_i(t+1) = \begin{cases} 1 \text{ if } c_{i-1}(t) + c_{i+1}(t) \text{ is odd} \\ 0 \text{ if } c_{i-1}(t) + c_{i+1}(t) \text{ is even} \end{cases}$$

This means that those cells with an odd number of live neighbours at time t turn themselves on at time $t + 1$, while those with an even number of live neighbours turn themselves off.

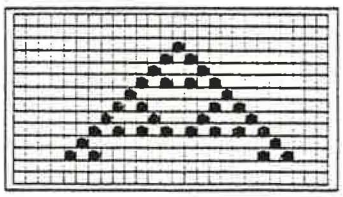


Figure 1

The easiest way to visualise how this cellular automaton evolves is to draw its successive states below one another (see Figure 1). Here, each row shows one further step in time. Black spots represent cells that are on, and white spots represent cells that are off. Other update rules have different behaviour. Rather than have special rules for the cells at the edge, most cellular automata "wrap around", so that the cell furthest left is adjacent to the cell furthest right.

The first person to study cellular automata was the mathematician John von Neumann, one of the founders of computer science. Von Neumann wanted to know if it were possible, even theoretically, for machines to reproduce; in other words, for some sort of robot to construct automatically an identical copy of itself. Rather than try to build such a robot using the vacuum-tube technology of his day, he chose to create a simple model universe, a rectangular, two-dimensional automaton. One cell state represented empty or unused cells; the other states (he eventually needed another 28) represented the components out of which he built his self-reproducing machine.

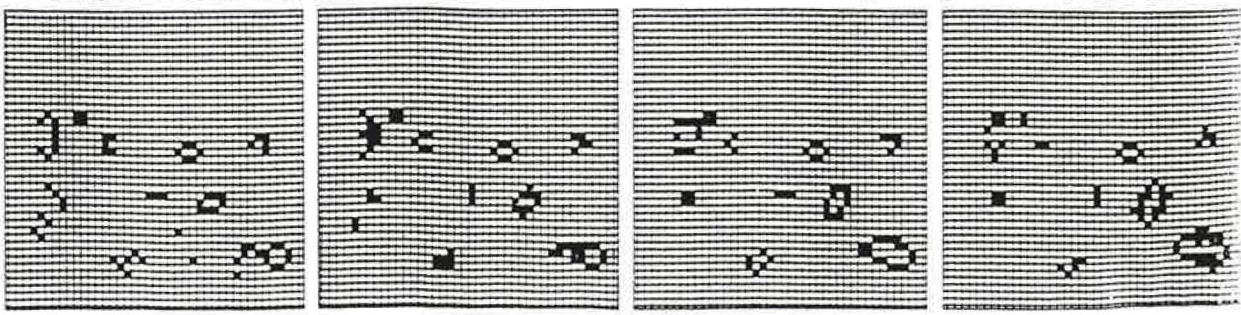
Controlled completely by the rules of the cellular automata, the machine (really just a pattern of cells in the cellular automaton) would extend an arm into a virgin portion of the universe, then slowly scan it back and forth, creating a copy of itself. Once the copy was complete, the original would retract its arm. The copy could then make a copy, and so on.

After von Neumann, cellular automata languished until 1970. In that year, John Conway, at Cambridge University, invented what has become the most famous cellular automaton of all, the "Game of Life". Conway was fascinated by the way in which a combination of a few simple rules could produce patterns that would expand, change shape, or die out unpredictably. He wanted to find the simplest possible set of rules that would give such interesting behaviour.

The rules he developed could not be simpler. Life is played on a rectangular grid, the same as von Neumann's original model universe. (Rectangular grids are probably popular more because graph paper is easy to obtain rather than because of any theoretical properties.) Cells in the grid are either alive or dead. If a cell is dead, but has exactly three live neighbours, it will come to life at the next tick of the clock. If a cell is alive, on the other hand, it will stay alive only if it has either two or three live neighbours.

Since 1970, people have written many programs to simulate Life. Some programmers have become addicted to the strange forms of Life that these simulations produce. Figure 2 shows a small universe of Life evolving. Notice the "glider" crawling toward the upper right corner, and the

Figure 2



The most famous cellular automation of all: Conway's universe of Life evolves on a computer screen

stationary "blinker" and "beehive" patterns near the centre. Larger and more complicated patterns include such things as "glider guns", which periodically fire gliders into space, and "puffer trains"—patterns that travel across the universe at a fixed speed leaving a chaotic cloud of "exhaust" in their wake.

In fact, the patterns produced by Life can be so complex that the cellular automaton has been shown to be equivalent to a Turing machine. A Turing machine is a simplified model of a computer made up of an infinitely long tape, on which symbols are recorded and erased, and a read/write head that moves backward and forward under the control of a very simple program. In the 1930s, the English mathematician Alan Turing showed that any computation that could be carried out by any conceivable computer could also be done (albeit perhaps more slowly) by his idealised model. Showing that Life could do everything a Turing machine could do was, therefore, equivalent to showing that given the right initial conditions, a game of Life could simulate any possible computer.

The most important consequence of this is that there is no way to discover the long-term behaviour of an arbitrary Life pattern other than by simulating it. If there were, we could apply the same technique to the other types of computer (including real electronic computers) known to be equivalent to Life. Turing and Kurt Gödel, the Austrian logician, proved these techniques did not exist. This constitutes a fundamental theorem of computer science known as the "halting problem".

The same sort of unpredictability has become a hot topic in physics during the past decade. Consider a system of the kind that the classical dynamics of the 18th and 19th centuries studied, such as a ball rolling down a plane. You can describe the behaviour of this kind of system using a differential equation that can be solved for any time t. Simply by changing the value of t, you can determine what the system looked like, or will look like, at any time.

Now consider the same ball cannoning back and forth on a snooker table. Where will the ball be after a half dozen bounces? It turns out that the answer is not calculable, practically. Any change, no matter how small, in the ball's initial position or velocity is magnified by each successive bounce. Its final position can vary enormously as a result of an unmeasurably small change in the initial conditions. The same phenomenon occurs in Life—changing a single cell in a universe of any size can have an arbitrarily large effect, or no effect at all.

Systems with this property are called "chaotic dynamical systems" because of the chaos that a small change can cause. Chaos of this kind is unrelated to quantum uncertainty—a rebounding snooker ball is much too large for quantum effects to be significant. Almost all real systems are chaotic, but we usually analyse them under a set of simplifying assumptions because chaos is so difficult to tackle mathematically.

Cellular automata are some of the simplest chaotic systems known. Even simple one-dimensional cellular automata, such as the one shown in Figure 1d, can exhibit chaotic behaviour. Interest in these properties sparked the current wave of interest in cellular automata. The spark itself was a series of papers by Stephen Wolfram, then at the Institute for Advanced Studies at Princeton.

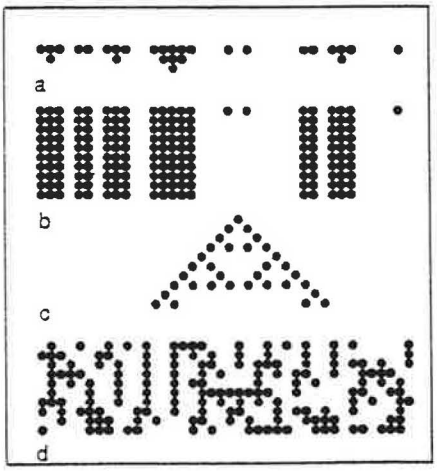
Wolfram investigated cellular automata with tools taken from both physics and computer science. Using the first approach, he analysed them as discrete approximations to continuous differential equations. This allowed him to use thermodynamic measures, such as entropy, to characterise their behaviour. Using the second approach, he described cellular automata as language-generating machines, producing patterns that conformed to the rules of formal grammars of different levels of complexity.

Wolfram classified cellular automata into four types: Type I, in which the pattern disappears with time (Figure 3a); Type II, in which the pattern evolves to a fixed finite size (Figure 3b); Type III, in which the pattern grows indefinitely at a fixed speed (Figure 3c); and Type IV, in which the pattern grows and contracts irregularly (Figure 3d).

Type IV includes those that show chaotic behaviour. Any small change in the initial configuration of cellular automata of Type IV can (but will not necessarily) propagate further through successive generations until every cell in the cellular automaton has been affected.

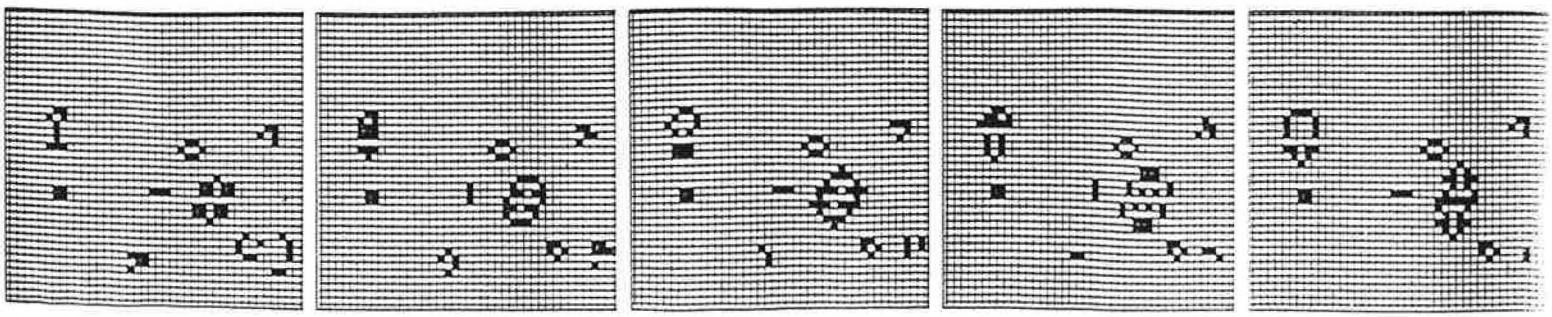
Type III cellular automata are interesting for a different reason. Cellular automata of this type often produce self-similar patterns known as "fractals". Figure 1c is an example of such a pattern. If we expand part of the figure, it has the same shape as the figure as a whole. Fractals are being used in fields as diverse as computer graphics and theoretical physics to describe the appearance of clouds and the way space folds on itself. As in chaotic dynamics, people interested in fractals are studying cellular automata because they are one of the simplest systems that can produce interesting behaviour.

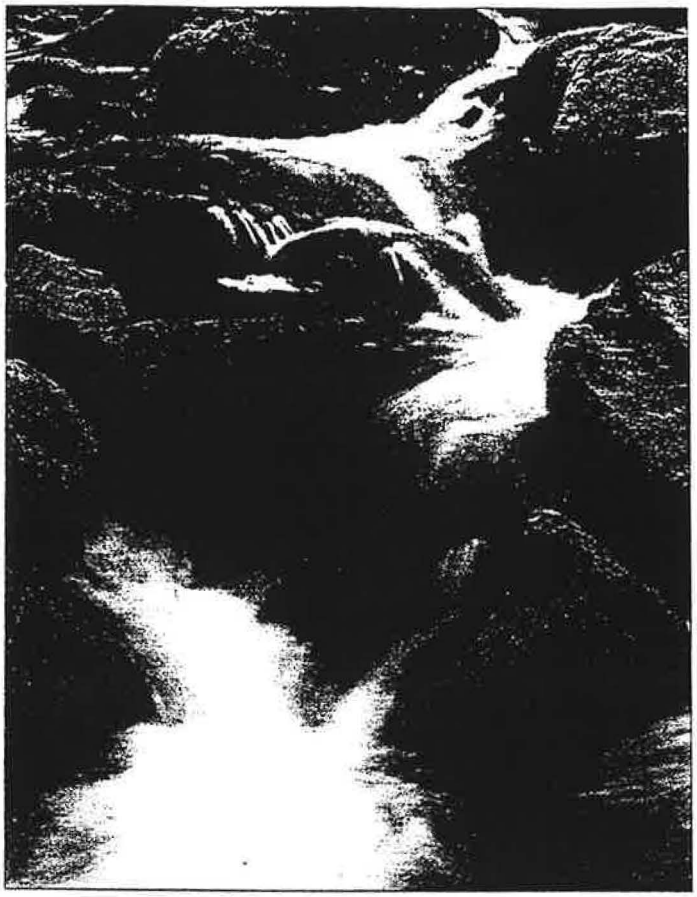
Although cellular automata are a fascinating subject in their own right, the emphasis of interest has shifted from their general properties to the possibility of using them to simulate a variety of physical processes such as fluid flow, diffusion, and crystal growth. Such simulations, generally known as



Bassano and Vandyk Studios

Cellular automata can behave like the idealised machine invented by Alan Turing





Tony Stone/Tony Craddock

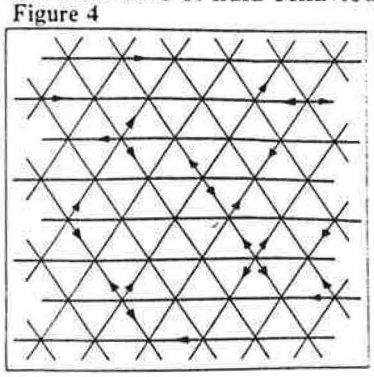
Cellular automata may be able to model the turbulent flow of liquids, such as this waterfall, on a computer screen

"lattice gases", are an example of the way science progresses by producing simple models of the phenomena it studies.

Consider a fluid such as whisky. At the microscopic level it is made up of many particles, often of different shapes and masses, moving in all conceivable directions with a wide spread of velocities. We can describe its macroscopic properties by averaging out the motion of the particles in each small volume of the fluid. This produces a set of differential equations known as the "Navier-Stokes" equations, which in principle gives us a complete description of the fluid's behaviour. Unfortunately, the Navier-Stokes equations are not solvable in any except the simplest situations. In order to find out how real fluids behave in real situations, scientists and engineers must use experiments, numerical approximations to the Navier-Stokes equations, or computer models.

Clearly, the simpler the model used in a computer simulation, the more quickly it will run (or the larger the model we can examine in a given time). This consideration led researchers to ask how many of the properties of real fluids are intrinsic to the nature of the fluid. After all, materials as different as honey and liquid hydrogen all obey more or less the same laws of fluid behaviour.

Figure 4



A lattice gas replaces the diversity of a real fluid with a regular grid, on which identical particles move step by step (Figure 4). Instead of the complex interactions of real molecules, a lattice gas handles collisions according to simple rules that respect only the simplest strictures of Newtonian physics. The conservation of mass demands that the same number of particles leave a collision as entered it; this forbids rules such as the first one shown in Figure 5a. Similarly, the conservation of momentum forbids the second rule in Figure 5b, because the vectors sum including speed and direction of the motions before and after the collision are different. The third rule of Figure 5c is not disallowed by physics, but by common sense —the outcome is the same as if the particles had not interacted at all, which is unlikely to produce interesting behaviour. Only the fourth rule of Figure 5d is both "legal" and useful.

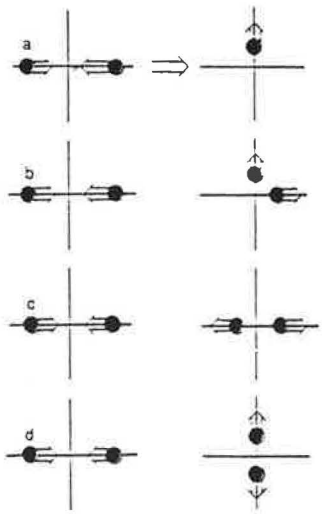
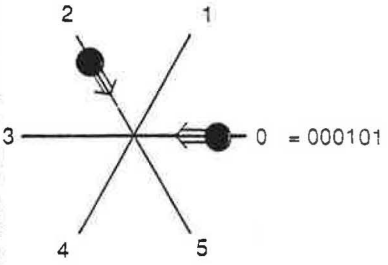


Figure 5

The simplest way to implement a lattice gas is to use a cellular automaton. Each grid point is replaced by a cell, whose state encodes the number and directions of incoming or outgoing particles (Figure 6). At each time step, every cell compares its state with that of its neighbours and decides what state it will be in next—how many of its particles will have moved elsewhere and how many new ones will have arrived. The number of possible situations is relatively small (2^6 = 64 in the hexagonal model), and can be reduced to the 13 shown in Figure 7, by noting that many situations are just rotations, or mirror images of one another.

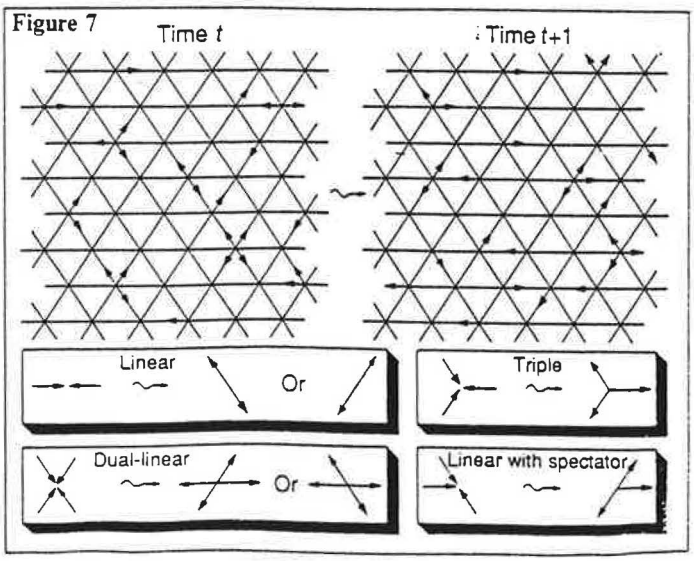
Figure 6



Once the grid is operating, we can calculate the properties of the fluid by averaging the motions of the particles over a large patch of the simulation to obtain an average momentum of flow for that part of the fluid. If this averaging process results in properties with the characteristics described by the Navier-Stokes equations, the underlying model is valid.

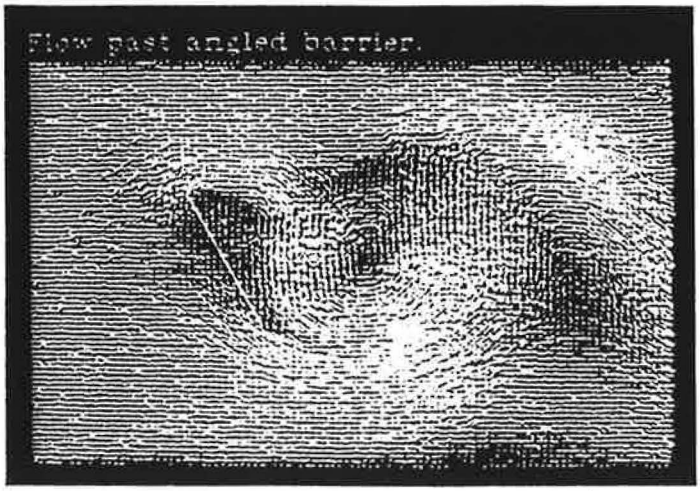
The computer graphic opposite shows a lattice-gas model that Brian Wylie, Richard Kenway, and W. D. McComb at the University of Edinburgh have developed. Each arrow shows the average momentum of an area with 50 cells on a

Figure 7



side. This simulation runs on a Meiko Computing Surface, a British-built supercomputer, that uses parallel processing. It has modelled such situations as flow through constrained channels, the way eddies are created and shed by an oblique barrier, and the formation of a vortex downstream from a pipe.

One advantage of lattice gases over more conventional techniques is the ease with which we can model irregular barriers. A barrier is represented in a lattice gas by a group of special cells through which particles cannot pass. These cells use a different set of update rules from normal cells, which cause incoming particles to reflect or bounce at an angle. This is much simpler than trying to express the shape of a complicated or irregular wave as an equation.



A computer simulation of fluid flowing past an angled barrier

In 1986, Uriel Frisch, Brosl Hasslacher, and Yves Pomeau at Los Alamos in New Mexico proved that you needed a hexagonal grid like the one in Figure 4. Earlier work by Pomeau, O. de Pazzis, and J. Hardy in the 1970s had shown that a square grid, like the one in Figure 4, lacked sufficient symmetry to reproduce hydrodynamic behaviour properly. Averaging particle behaviour over large lattice areas failed to produce the Navier-Stokes equations as required. They could clearly see the effect of this in simulations, in which barriers cast shadows of fluid-free areas rather than causing eddies to form.

A hexagonal grid, on the other hand, possesses the necessary symmetries. By the time Frisch, Hasslacher, and Pomeau showed this, the computing power required to do large simulations had become more widely available, so more researchers could start to investigate these models.

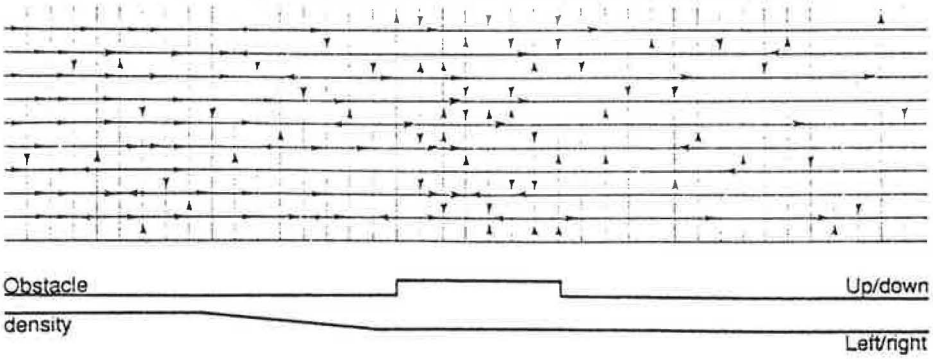
Even after the microscopic model has been shown to be valid, researchers still have many choices to make when implementing lattice gases. Some situations, for example, can have several different outcomes, as shown in Figure 7. A simulation could use the same one every time (a "deterministic" model), or select among the possibilities at random (a "non-deterministic" model). Although the latter is more realistic, it is also much more expensive, because generating the random number needed may take more time than updating the cell. One way to obtain the best of both worlds is to use a "pseudo-deterministic" model in which the computer selects different outcomes based on some non-physical parameter of the system, such as whether the time step is even or odd.

One challenge for lattice gas models is how to use them in three dimensions. No regular three-dimensional grid has enough symmetry to simulate lattice-gas fluids. There is, however, a four-dimensional grid called a face-centred hypercubic which is symmetrical enough. We can simulate flow in three dimensions by using a four-dimensional grid that is extremely thin in its fourth dimension, and then projecting the results back into three dimensions.

A more important, and still unsolved, problem is how to model compressible fluids. The real Navier-Stokes equations contain several terms, one of which describes the temperature of the fluid at any point. This term does not appear when we derive the macroscopic behaviour of lattice gas models. The major effect of compression on a fluid is to change the fluid's temperature, so the absence of this term means that cellular automata can model only incompressible fluids properly.

One advantage of lattice gases, which compensates for these problems, is that they are well suited to being implemented on silicon chips. Because the behaviour of each lattice site in a cellular automaton is so simple, engineers can put hundreds of such lattice sites on a single chip using very-large-scale integration (VLSI). Hundreds or thousands of these chips can then be wired together to produce a "numerical wind tunnel", a computer whose only purpose is to simulate aerodynamics. Specialised hardware of this kind has already been built, for example, by Steve Kugelmass and Kenneth Steiglitz at Princeton University. Such computers will be indispensable tools in the design of the next generation of aircraft and spacecraft. An interest in cellular automata continues to grow, it seems that von Neumann's self-replicating machines are here to stay. □

Particles, obstacles and models of diffusion



PHYSICISTS are employing lattice gas models in other areas as well as fluid dynamics. In the United States, Bruce Boghosian and C. David Levermore have been using them to model diffusion.

One of their models uses a rectangular lattice, and rules that conserve the number of particles moving parallel to each axis independently. While particles move in all directions, the numbers of particles moving left/right and up/down are the same after collisions as they were before. In effect, a given particle spends its life moving parallel to either the x or y axis.

As a result, the rate of diffusion along one axis depends on the density of particles moving along the other axis. In particular, by varying the number of particles moving parallel to the y axis, we can "tune" the rate of diffusion of particles moving parallel to the x axis. This makes it easy to simulate materials that are not homogeneous.

A long, narrow grid is used for this (see above). The diffusing particles moving left and right, while the particles moving up and down exist only to act as obstacles. Increasing the number of obstacle particles in a particular region of the grid makes diffusing through that part of the grid harder. The same technique can model two-dimensional diffusion by allowing some particles to move freely in the xy plane, and others to move only parallel to the z axis. □

Greg Wilson works in the Edinburgh Concurrent Supercomputer Project at the University of Edinburgh.