



A CRASH COURSE ON COMPUTING

By Bruce Boghosian

Collision-Based Computing, Andrew Adamatzky, ed., Springer, 2002, US\$89.99

THE BOOK *COLLISION-BASED COMPUTING*, EDITED BY ANDREW ADAMATZKY, DESCRIBES HOW THE PHENOMENON OF COMPUTATION CAN ARISE NATURALLY IN A PHYSICAL MEDIUM BY MEANS OF STRUCTURES THAT MOVE AND INTERACT WITH ONE

another. These structures might be anything from electrical pulses to billiard balls to patterns of chemical concentration. The book explores the ragged interface between physics and computation, touching on both the impact of physics on computation and the impact of computation on physics. To explain the book, I'll provide a "crash course" on recent developments in the theory of computation, beginning with a description of the discovery and the impact of John Horton Conway's *Game of Life* in 1970.

The Game of Life

In October 1970, the Mathematical Games column in *Scientific American* published the first of a series of articles on a remarkable dynamical system invented by Conway, a mathematician, which has come to be called "the game of life" (GOL).¹ GOL is an example of a *cellular automaton*, a term first coined by the mathematician and pioneer of computer science, John von Neumann, to characterize a dynamical system that is discrete in space, time, and state. Discreteness in space means that the dynamics occur on a lattice of cells; in time, it means that the cells update their states in a sequence of time steps; and in state, it

means that each site's state could be described with only a finite number of bits of information. At every tick of the clock, each cell transitions to a new state that depends only on its current state and those of its neighbors according to a deterministic rule. The very same rule is applied to each cell simultaneously.

Although this definition of cellular automata is commonly used and technically correct, some vague and unspoken rules should be kept in mind. First, the number of bits of state at each site should be small. Second, the rule used to update the states should be simple. After all, an explicit finite-difference algorithm is discrete in space and time, and if it is implemented with IEEE floating-point numbers, it is also discrete in state. If it has periodic boundary conditions, it also has a rule that is applied simultaneously at every spatial location. Nevertheless, this isn't usually called a cellular automaton because it's not simple. The IEEE implementation of even a single floating-point number has 64 bits: 52 for the mantissa, 11 for the exponent, and one sign bit. Written in terms of these constituent bits, a finite-difference algorithm appears exceedingly complicated.

By contrast, Conway's GOL is a paragon of simplicity. It employs a two-dimensional Cartesian lattice, each cell of which contains only a single bit of state. Following the metaphor of life, each cell is "alive" (its bit is equal to one) or "dead" (its bit is equal to zero). The update rule is also very simple and depends only on the cell's present state and those of its eight neighbors (at the four compass points and the diagonal directions). If a cell is dead and has exactly three living neighbors, it becomes alive; abusing the metaphor further, we call this event a "birth." If a cell is alive, it will remain alive if it has either two or three living neighbors. If it has fewer than two living neighbors, it dies of "loneliness;" if it has four or more living neighbors, it dies of "overcrowding."

This rule is so simple a child can understand it—indeed, I was a child when Gardner's GOL articles first appeared, and I can vividly remember consuming vast quantities of graph paper as I experimented with various initial configurations. Of course, it was impossible to keep up with the university researchers who had access to PDP-7 computers. Later in life, I taught the GOL rules to my own children, but by then computers had become common and powerful enough to take all the mystery out of the endeavor.

Like an initial-value problem for a partial differential equation, a cellular automaton describes the evolution of a spatially extended state from some

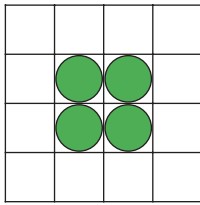


Figure 1. Stable two-by-two “block” of live sites. Each of the four live sites has exactly three neighbors.

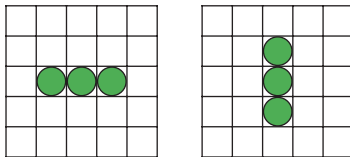


Figure 2. Two successive states of a “blinker.” The three live sites in a row will rotate by 90 degrees in (a) one time step and (b) back again in the next.

specified initial condition. Given the parsimony of Conway’s GOL as compared to, say, a forward-time centered-space explicit finite-difference algorithm for the heat equation, it might come as a surprise that the former is capable of far more complicated and fascinating behavior than the latter.

At the outset, it’s easy to find the stable states: a two-by-two block of live sites with no other live sites near it, for example, will survive indefinitely because each of the four live sites—and none of the surrounding dead sites—has exactly three neighbors (see Figure 1). We also see that oscillatory states, or *limit cycles* in state space, could exist; three live sites in a row, with no other live sites nearby, form a “blinker” that will rotate by 90 degrees in one time step and back again in the next (see Figure 2). So far, no surprises: stable solutions and limit cycles are familiar enough from the study of ordinary and partial differential equations.

The “glider” configuration, consisting of five live sites, is more captivat-

ing. As Figure 3 shows, it transforms in four generations to a copy of itself shifted diagonally by one unit in each direction. When GOL is simulated on a computer, the glider appears to crawl diagonally across the screen. This movement is reminiscent of a soliton solution to a nonlinear partial differential equation, which maintains its shape as it translates. To maintain historical perspective here, keep in mind that the inverse scattering method of solving the Korteweg deVries equation was only a few years old when GOL was discovered.^{2,3} Suddenly, the dynamical system that even a child could understand was exhibiting behavior at the very frontiers of research on spatially extended dynamical systems.

A zoo of stable, oscillating, and even gliding objects soon followed. Some of these were discovered as by-products of evolution from random initial conditions, but others were built by people who had learned to “engineer” objects for various purposes in the GOL universe. I use this last word deliberately because it became impossible not to think of GOL as a world of its own, with its own laws of nature. Early practitioners likened gliders and other translating objects to particles moving about, interacting with each other and with stationary objects.

Gliders were also likened to electrical pulses in a digital circuit. With the latter metaphor in mind, GOL engineers such as William Gosper at MIT constructed logical gates so that they could build circuits to perform computations with gliders representing data—one level of abstraction above the actual bits in the cells themselves. It requires only NAND gates and a clock that can produce a regular stream of bits to build a computer. Gosper discovered the latter ingredient—a glider gun that produces a stream of gliders at

regular intervals—after Conway offered a \$50 reward to the first person who could demonstrate initial conditions for which the number of live cells grows without bound.

The construction of a computer within the GOL universe might seem like a frivolous exercise, but it demonstrates a critically important property of complex dynamical systems—something that was hitherto unsuspected in the context of partial differential equations. Computers have a very special property: it’s provably impossible to know their final time-asymptotic state for arbitrary initial conditions. It’s impossible to know, for example, whether they’ll halt for arbitrary input. The attempt to find an algorithm to determine whether a given computer with a given input will halt was called the *halting problem*, and Alan Turing demonstrated the impossibility of such an algorithm in the 1930s. The halting problem is closely related to Gödel’s incompleteness theorem, because the question of any number theory statement’s truth or falsity can quickly be transformed into the question of whether a certain computer program will halt. It isn’t terribly difficult, for example, to construct a computer program that will halt if and only if it finds a counterexample to Goldbach’s conjecture. In principle, it’s possible to set up GOL on a sufficiently large grid with initial conditions configured to run any such program. In other words, GOL is capable of universal computation, and so it’s just as impossible to know its time-asymptotic state for arbitrary initial conditions as it is to prove an arbitrary statement of number theory.

To be sure, partial differential equations exist the determination of whose time-asymptotic state for arbitrary initial conditions has eluded the best ef-

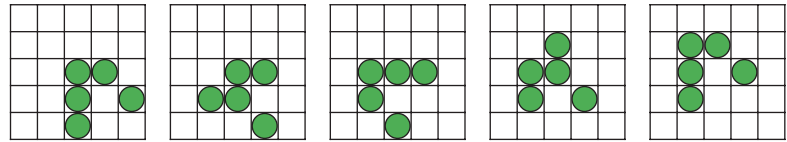


Figure 3. Five successive states of a “glider.” The sites can transform in four generations to a copy of the glider, shifted diagonally by one unit in each direction.

forts of generations of mathematicians. The 3D Euler and Navier-Stokes equations of fluid dynamics leap to mind here, but we don’t know if the difficulty arises because these equations could somehow be used to perform a computation, or if we just haven’t been smart enough to find a general theory for their asymptotics. For GOL, though, there is no question: a general theory of its time asymptotics can’t exist.

Artificial Intelligence

Conway has even speculated that the GOL universe has sufficient complexity to sustain artificial life forms, complete with evolution driven by natural selection (see <http://plus.maths.org/issue20/features/conway/>). After all, we can build universal computers in this universe, and Strong AI advocates maintain that this is all that’s necessary for artificial life to flourish. Any finite computer program for artificial life could be implemented within GOL, but the initial state would likely be a large and unwieldy affair. With random initial conditions on a 2D grid, however, it would exist somewhere with probability unity. GOL is likely to be a brutal world for these organisms, littered with stable, oscillating, and gliding structures that could wreak havoc with a carefully designed artificial life form. Nevertheless, some will be better than others at surviving for long periods of time, and Darwinian evolution will follow. Perhaps such beasts might eventually explore their world well enough to develop an internal representation of GOL’s underlying rules, thereby providing us with a metaphor for our efforts to probe the physical rules governing our own universe.

All this from a simple rule that even a child can understand....

If you think that this is fascinating,

you’re not alone. Type the keywords “Conway” and “Game of Life” into Google, and you’ll find nearly 40,000 hits related to the cellular automaton. People who develop software arsenals to study various aspects of GOL participate in active electronic discussion lists—some work as mathematicians and computer scientists, but others are amateurs, hobbyists, and diletantes who are captivated by the beauty of it all. My strong suspicion is that the CIOs of many a Fortune 500 company would cringe if they learned how many of their cycles were being siphoned off to hunt for, say, the smallest possible “Garden of Eden” state—a state that has no predecessor according to the GOL rules. (The latest record for this, set in June 2004, is a state with 72 live cells in a 12×11 rectangle; see <http://wwwhomes.uni-bielefeld.de/achim/gol.html>.)

Given that GOL has demonstrated that universal computation is a property dynamical systems might have, we should ask if the laws of nature in our own universe share that property. We can build computers in our universe, and the workings of those computers obey our laws of nature so, yes, our laws of nature are capable of computation. In fact, as Edward Fredkin, Norman Margolus, and Tomaso Toffoli showed more than 20 years ago, even conservative mechanical systems as simple as colliding ideal billiard balls can be used to construct a universal computer. This work both expanded the theory of computation and gave us a new appreciation for the limitations of mathematical analysis of dynamical systems.

Since then, we’ve learned that dy-

namical systems as diverse as reaction-diffusion equations and nonlinear wave equations are capable of computation. Computation can happen when you pour chemicals in a Petri dish or when you fire light pulses down a fiber optic cable. Indeed, it seems that sometimes computation can happen even when you don’t intend it to happen.

The Book

Collision-Based Computing is a collection of some of the most important papers in this field, together with new work. Part I is entitled “Twenty Years Ago,” and it reprints original papers, including a hitherto unpublished project description sent to DARPA in 1978, called “Design Principles for Achieving High-Performance Submicron Digital Technologies.” This paper outlined the idea of conservative logic.

Because most of the logical gates commonly used in computer circuitry accepted two bits of input and yielded one bit of output (for example, the NAND, OR, XOR, and AND gates), they weren’t reversible; you couldn’t determine the input from the output. If you try to fix this by adding one more bit of input and restricting attention to reversible two-bit-in/two-bit-out gates, you’ll find that they aren’t capable of universal computation. For this reason, a school of thought maintained that conservative universal computation was impossible. The idea was that some information had to be destroyed in any universal computer, that the loss of this information gave rise to a corresponding increase in entropy, and that the increase in entropy at finite tem-

perature meant that computers must always dissipate heat.

Fredkin and Toffoli pointed out that they could construct reversible three-bit-in/three-bit-out gates capable of universal computation. Computers didn't have to dissipate heat after all. In a subsequent paper, also reprinted in Adamatzky's book, they described the physical implementation of these gates using colliding billiard balls. Margolus then constructed a reversible cellular automaton rule that implemented the billiard-ball style of computation in a

sion of it with 11 states per site, which he shows is minimal.

The idea of making models of computation as physically realistic as possible is a pervasive theme throughout this book. Kenichi Morita and his coauthors describe reversible universal computation with physical models that have a conservation law. Marianne Delorme and Jacques Mazoyer then give an information-theoretical description of what it means for a cellular automaton to support a signal. Andrew Wuensche presents a general

and NOT gates in detail, as well as how to use them with a glider gun to construct a universal computer in the GOL universe.

Though this book is now more than two years old, it remains a fine collection of interesting review papers that could be used to bring a new graduate student up to speed in the field. It will be of interest to computer scientists who care about the frontiers of the theory of computation, as well as to physicists and applied mathematicians who are interested in the impact of the theory of computation on their fields. At the same time, much of the book is perfectly accessible to undergraduates and the many hobbyists who have both dabbled in this field and made substantial contributions to it. I believe that this book, along with Adamatzky's earlier book, *Computing in Nonlinear Media and Automata Collectives*, is destined to become a standard reference in the field.

SE

Type the keywords 'Conway' and 'Game of Life' into Google, and you'll find nearly 40,000 hits related to the cellular automaton.

natural way, which was the first example of a reversible cellular automaton capable of universal computation. (Remember, GOL isn't reversible: many different states can simply die away leaving no trace at all, for example, so you can't recover earlier states from later ones.)

Part II of the book is entitled "The Present and the Future," and it presents current work in this still active field. The first paper in this section—written by Margolus—shows that you can employ more realistic potential functions without discontinuities and without losing the property of universal computation. In the effort to make physical models of computations as realistic as possible, he also discusses relativistically invariant physical models of computation. Jérôme Durand-Lose describes the billiard-ball model of computation in great detail and presents a block cellular automaton ver-

method for finding glider-like objects in cellular automata.

The book also features papers on computing with solitons by Mariusz Jakubowski and his coauthors, Pawel Siwak, and Steve Blair and Kelvin Wagner. Adamatzky himself contributes a paper on new media for collision-based computing. This includes a fascinating description of computing in liquid crystals in which information is stored in topological defects.

Perhaps fittingly, the book ends with a paper by Jean-Philippe Renard on universal computation in GOL. It's one thing to argue that you could build a universal computer with the components available in the GOL universe—it isn't unlike arguing that given a saw, a hammer, and a sufficient supply of nails and trees, you could build a house—but Renard sets out to show us how to build the house. Using gliders as "bits," he describes AND, OR,

References

1. M. Gardner, "The Fantastic Combinations of John Conway's New Solitaire Game 'Life,'" *Scientific Am.*, vol. 223, Oct. 1970, pp. 120–123.
2. C.S. Gardner et al., "Method for Solving the Korteweg-de Vries Equation," *Physical Rev. Letters*, vol. 19, Nov. 1967, pp. 1095–1097.
3. P. Lax, "Integrals of Nonlinear Evolution Equations and Solitary Waves," *Comm. Pure Applied Mathematics*, vol. 21, 1968, pp. 467–490.

Bruce Boghosian is professor of mathematics and an adjunct professor of computer science at Tufts University. His research interests include theoretical and computational fluid dynamics, dynamical systems, and quantum computing. Boghosian has a PhD in applied science from the University of California, Davis. Contact him at bruce.boghosian@tufts.edu.