Undecidable dynamics

Charles H. Bennett

ALLUDING to the effects on twentieth-century physics of quantum mechanics and general relativity, the cosmologist John Wheeler once remarked that the world is simpler and stranger than we can imagine. An analogous dose of strange simplicity was delivered to mathematics in the 1930s by the discoveries of Gödel, Turing and their contemporaries in the theory of proof and computation. Yet there have been surprisingly few applications of these ideas to physics. In particular, the phenomenon of 'undecidability' has not asserted itself in any simple, uncontrived physical setting the way that deterministic chaos, a far weaker property, has. (For a dynamical system to be chaotic means that it exponentially amplifies ignorance of its initial condition; for it to be undecidable means that essential aspects of its long-term behaviour — such as whether a trajectory ever enters a certain region — though determined, are unpredictable even from total knowledge of the initial condition.) But C. Moore

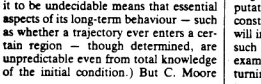

FIG. 1 A universal Turing machine consists of an infinite passive tape and a finite active read/write head. It can be programmed, through the initial state of its tape, to perform any computation.

has now come closer than any to a simple dynamical representation of undecidibility.

The theory of computation encompasses any situation in which discrete symbols are manipulated according to a fixed program or algorithm. A distinctive feature of computation is the ability to program an open-ended search for something that may or may not exist, such as a counterexample to Fermat's Last Theorem, which asserts that the equation $x^n + y^n = z^n$ has no solutions in positive integers for $n > 2$. If a counterexample exists, the computation will find it, but if it does not exist, the computation will continue forever without finding it. The lack of any general procedure for deciding *a priori* whether computations terminate, called the 'unsolvability of the halting problem', is the central negative result of computability theory. Not only termination, but any other global property of computations that might be used to signal the outcome of an open-ended search, such as whether a certain bit is ever turned on, is similarly algorithmically undecidable.

The central positive result of computability theory is the existence of 'universal computers', which can be programmed to simulate any other computer and perform any digital computation. Computational universality manifests itself in the well known ability of real computers (other than specialized ones like those in digital watches) to simulate one another, so that they differ in speed and convenience, but not in the range of computations they can be made to perform. By extension, a discrete or continuous dynamical system is called computationally universal if it can be programmed through its initial conditions to perform any digital computation.

For example, the cellular automaton game Life — in which the state ('alive' or 'dead') of each cell in an infinite array of cells changes according to the states of its neighbours — has been shown to be computationally universal. One can therefore construct an initial condition for it that will implement an arbitrary computation, such as conducting a search for counterexamples to Fermat's Last Theorem and turning on a designated cell when a counterexample is found.

Universality and undecidability are closely related: roughly speaking, if a universal computer could see into the future well enough to solve its own halting problem, it could be programmed to contradict itself, halting only if it foresaw that it would fail to halt. We are thus left with a tantalizing situation. On the one hand, the global map of any computationally universal process — the iterative limit of its operations — is an infinitely valuable microcosm of all iterative processes and all cause–effect relations that can be demonstrated by deductive logic or numerical simulation. On the other, because of the unsolvability of the halting problem, this same global map cannot be computed directly. It can only be approximated as the limit of an iterative process that converges with uncomputable slowness.

Computational universality was originally demonstrated for a very simple type of ideal computer, the Turing machine, which consists of an infinite passive tape, on which data is stored in the form of binary digits 0 and 1, and a finite active head which shifts the tape back and forth manipulating the data one bit at a time (Fig. 1). It is this type of computer which Moore has recently succeeded in embedding in a continuous dynamical system. Ideally, as Moore points out, one would like to prove computational universality, and undecidability, for some famous problem in continuum dynamics, like the three-body problem — the calculation of the orbits for three interacting masses. Moore's system, while still simple, is less natural: it corresponds to the motion of a single classical particle in a three-dimensional box containing a finite number of plane and parabolic mirrors.

Moore's construction generalizes the ability (Fig. 2) of simple chaotic dynamical processes to split a parallel bundle of trajectories in two, deform and reunite the parts, then feed them back into the original bundle in such a way that trajectories are continually pulled apart in one direction (here horizontally) and squeezed together in another (here vertically). Associated with any such process is a certain fractal set of initial conditions shown in the enlarged cross-section of the bundle and called a Cantor set. This set

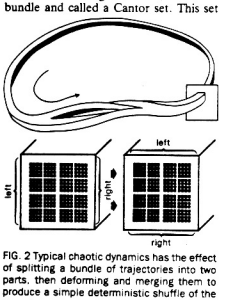

FIG. 2 Typical chaotic dynamics has the effect of splitting a bundle of trajectories into two parts, then deforming and merging them to produce a simple deterministic shuffle of the associated Cantor set visible in the enlarged cross-section. Moore's generalization of this splits the bundle into several dozen parts, then deforms and merges them to produce a more complicated shuffle equivalent to the action of a Turing machine

has the form of a solid square from which a vertical and a horizontal cut have been removed leaving four smaller squares, each of which is further subdivided in the same way *ad infinitum* until no solid contiguous pieces remain. The dynamics has the effect of performing a rather simple deterministic shuffle of all the infinitely many points in the Cantor set, so that each point in the left half of the Cantor set, after one turn of dynamics, ends up as a corresponding point in the top half, and each point in the right half ends up in the bottom half.

This shuffle can be described more abstractly as a repeated left-shifting of an infinite binary sequence whose left and right sides represent, respectively, a point's vertical and horizontal address within the Cantor set. Left-shifting a sequence without changing it is just a rather trivial Turing-machine computation, and Moore's particle-and-mirror system manages to generalize the dynamics enough to implement a full range of Turing-machine actions needed for universality. Plane mirrors are used to split

and merge trajectory bundles to implement reading and writing, and the parabolic mirrors are used to squeeze and stretch the bundles to implement shifting.

Moore's is the most economical dynamical model of computation as it uses just three dynamical variables (the coordinates of a single particle) to encode the entire logical state of the computation. But, because logically distinct states are packed arbitrarily close together in physical space, Moore's dynamics cannot tolerate any amount of disturbance, however small, by external noise.

A less economical, but otherwise far simpler and more straightforward embedding of computation in dynamics is provided by Fredkin's billiard-ball model[2], which uses one ideally hard, frictionless billiard ball to represent each bit of data, and performs all necessary logic operations by causing the balls to collide with one another. Like Moore's, this model is reversible: ideally its operation would consume no energy, and reversing the velocities of all the balls (or Moore's one ball) would cause the motion to run backward, exactly undoing the computation.

The differential equations describing a real electronic computer provide another dynamical model at the same level of economy (a few dynamical variables — voltages and charges — per bit). Unlike the preceding models, an electronic computer can recover from small disturbances due to external noise: the ability to do so depends both on having enough dynamical variables to keep logically distinct trajectories physically far apart, and also on the equations' irreversibility, manifested in a real computer's consumption of electricity and production of heat.

At the opposite extreme from Moore's is a model sketched by Omohundro[3], which simulates an arbitrary cellular automaton by a set of space- and time-independent partial differential equations, thereby using infinitely many dynamical variables to encode each bit. Like a real computer, it is irreversible and can recover from small disturbances due to noise; combining it with recent error-correcting cellular automata[4,5] should yield models also able to recover from the large but rare disturbances characteristic of natural noise sources. Despite their infinitude of dynamical variables, partial differential equations provide the most natural mathematical model for processes occurring in uniform physical space-time, such as fluid flow and wave propagation. So, models of this type hold out the hope that some totally structureless homogeneous medium — an exotic chemical brew, say — might have universal computing abilities, just as simpler brews have the ability to crystallize or to generate spontaneous chemical waves and oscillations. □

Charles H. Bennett is at the IBM Research Center, Yorktown Heights, New York 10598, USA.

1. Moore, C. *Phys. Rev. Lett.* **64**, 2354–2357 (1990).
2. Fredkin, E. & Toffoli, T. *Int. J. theor. Phys.* **21**, 219–253, (1982).
3. Omohundro, S. *Physica* **10D**, 128–134 (1984).
4. Gacs, P. & Reif, J. *Proc. 17th ACM Symp. Theory Computing*, 388–395 (1985).
5. Gacs, P. *J. Computer System Sci.* **32**, 15–78 (1986).