the process, he thought, they were cutting across the traditional boundaries in a way they hadn't done in years. Maybe centuries.

One of their inspirations, ironically enough, seemed to be molecular biology. That's not the sort of thing that most people would expect a weapons laboratory to be interested in. But in fact, says Cowan, physicists have been deeply involved with molecular biology from the beginning. Many of the pioneers in the field had actually started out as physicists; one of their big motivations to switch was a slim volume entitled *What Is Life?*, a series of provocative speculations about the physical and chemical basis of life published in 1944 by the Austrian physicist Erwin Schrödinger, a coinventor of quantum mechanics. (Having fled from Hitler, Schrödinger spent the war safely ensconced in Dublin.) One of those who was influenced by the book was Francis Crick, who deduced the molecular structure of DNA along with James Watson in 1953—using data obtained from x-ray crystallography, a kind of submicroscopic imaging technique developed by physicists decades earlier. Crick, in fact, had originally trained as an experimental physicist. George Gamow, a Hungarian theoretical physicist who was one of the original proponents of the Big Bang theory of the origin of the universe, became intensely interested in the structure of the genetic code in the early 1950s and helped inspire still more physicists into the field. "The first really perceptive lecture I heard on the subject was by Gamow," says Cowan.

Molecular biology had fascinated him ever since, he says, especially after the discovery of recombinant DNA technology in the early 1970s gave biologists the power to analyze and manipulate life forms almost molecule by molecule. So when he became head of research at the laboratory in 1978, he had quickly thrown his support behind a major research initiative in the field—officially to study radiation damage in cells, but actually to get Los Alamos involved in molecular biology on a broader front. It was a particularly good time to do so, he recalls. Under director Harold Agnew, Los Alamos had nearly doubled its size in the 1970s, and had opened itself up to much more nonclassified basic and applied research. Cowan's emphasis on molecular biology fit right in. And that program, in turn, had had a tremendous impact on people's thinking at the laboratory. Especially his.

"Almost by definition," he says, "the physical sciences are fields characterized by conceptual elegance and analytical simplicity. So you make a virtue of that and avoid the other stuff." Indeed, physicists are notorious for curling their lips at "soft" sciences like sociology or psychology, which try to grapple with real-world complexity. But then here came molecular

biology, which described incredibly complicated living systems that were nonetheless governed by deep principles. "Once you're in a partnership with biology," says Cowan, "you give up that elegance, you give up that simplicity. You're *messy*. And from there it's so much easier to start diffusing into economics and social issues. Once you're partially immersed, you might as well start swimming."

But at the same time, says Cowan, scientists were also beginning to think about more and more complex systems simply because they *could* think about them. When you're stuck with solving mathematical equations by paper and pencil, how many variables can you handle without bogging down? Three? Four? But when you have enough computer power, you can handle as many variables as you like. And by the early 1980s, computers were everywhere. Personal computers were booming. Scientists were loading up their desktops with high-powered graphics workstations. And the big corporate and national labs were sprouting supercomputers like mushrooms. Suddenly, hairy equations with zillions of variables didn't look quite so hairy anymore. Nor did it seem quite so impossible to drink from the firehose of data. Columns of figures and miles of data tapes could be transformed into color-coded maps of crop yields or of oil-bearing strata lying under miles of rock. "Computers," says Cowan with considerable understatement, "are great bookkeeping machines."

But they could also be much more than that. Properly programmed, computers could become entire, self-contained worlds, which scientists could explore in ways that vastly enriched their understanding of the real world. In fact, computer simulation had become so powerful by the 1980s that some people were beginning to talk about it as a "third form of science," standing halfway between theory and experiment. A computer simulation of a thunderstorm, for example, would be like a theory because nothing would exist inside the computer but the equations describing sunlight, wind, and water vapor. But the simulation would also be like an experiment, because those equations are far too complicated to solve by hand. So the scientists watching the simulated thunderstorm on their computer screens would see their equations unfold in patterns they might never have predicted. Even very simple equations can sometimes produce astonishing behavior. The mathematics of a thunderstorm actually describes how each puff of air pushes on its neighbors, how each bit of water vapor condenses and evaporates, and other such small-scale matters; there's nothing that explicitly talks about "a rising column of air with rain freezing into hailstones" or "a cold, rainy downdraft bursting from the bottom of the cloud and spreading along the ground." But when the computer integrates those

equations over miles of space and hours of time, that is exactly the behavior they produce. Furthermore, that very fact allows the scientists to experiment with their computer models in ways that they could never do in the real world. What really causes these updrafts and downdrafts? How do they change when I vary the temperature and humidity? Which factors are really important to the dynamics of this storm, and which aren't? And are the same factors equally important in other storms?

By the beginning of the 1980s, says Cowan, such numerical experiments had become almost commonplace. The behavior of a new aircraft design in flight, the turbulent flow of interstellar gas into the maw of a black hole, the formation of galaxies in the aftermath of the Big Bang—at least among physical scientists, he says, the whole idea of computer simulation was becoming more and more accepted. "So you could begin to think about tackling *very* complex systems."

But the fascination with complexity went still deeper than that, says Cowan. In part because of their computer simulations, and in part because of new mathematical insights, physicists had begun to realize by the early 1980s that a lot of messy, complicated systems could be described by a powerful theory known as "nonlinear dynamics." And in the process, they had been forced to face up to a disconcerting fact: the whole really can be greater than the sum of its parts.

Now, for most people that fact sounds pretty obvious. It was disconcerting for the physicists only because they had spent the past 300 years having a love affair with linear systems—in which the whole is precisely *equal* to the sum of its parts. In fairness, they had had plenty of reason to feel this way. If a system is precisely equal to the sum of its parts, then each component is free to do its own thing regardless of what's happening elsewhere. And that tends to make the mathematics relatively easy to analyze. (The name "linear" refers to the fact that if you plot such an equation on graph paper, the plot is a straight line.) Besides, an awful lot of nature does seem to work that way. Sound is a linear system, which is why we can hear an oboe playing over its string accompaniment and recognize them both. The sound waves intermingle and yet retain their separate identities. Light is also a linear system, which is why you can still see the *Walk/Don't Walk* sign across the street even on a sunny day: the light rays bouncing from the sign to your eyes are *not* smashed to the ground by sunlight streaming down from above. The various light rays operate independently, passing right through each other as if nothing were there. In some ways even the economy is a linear system, in the sense that small economic agents can act independently. When someone buys a newspaper at the corner drugstore, for

example, it has no effect on your decision to buy a tube of toothpaste at the supermarket.

However, it's also true that a lot of nature is *not* linear—including most of what's really interesting in the world. Our brains certainly aren't linear: even though the sound of an oboe and the sound of a string section may be independent when they enter your ear, the emotional impact of both sounds together may be very much greater than either one alone. (This is what keeps symphony orchestras in business.) Nor is the economy really linear. Millions of individual decisions to buy or not to buy can reinforce each other, creating a boom or a recession. And that economic climate can then feed back to shape the very buying decisions that produced it. Indeed, except for the very simplest physical systems, virtually everything and everybody in the world is caught up in a vast, nonlinear web of incentives and constraints and connections. The slightest change in one place causes tremors everywhere else. We can't help but disturb the universe, as T. S. Eliot almost said. The whole is almost always equal to a good deal more than the sum of its parts. And the mathematical expression of that property—to the extent that such systems can be described by mathematics at all—is a *nonlinear* equation: one whose graph is curvy.

Nonlinear equations are notoriously difficult to solve by hand, which is why scientists tried to avoid them for so long. But that is precisely where computers came in. As soon as scientists started playing with these machines back in the 1950s and 1960s, they realized that a computer couldn't care less about linear versus nonlinear. It would just grind out the solution either way. And as they started to take advantage of that fact, applying that computer power to more and more kinds of nonlinear equations, they began to find strange, wonderful behaviors that their experience with linear systems had never prepared them for.

The passage of a water wave down a shallow canal, for example, turned out to have profound connections to certain subtle dynamics in quantum field theory: both were examples of isolated, self-sustaining pulses of energy called solitons. The Great Red Spot on Jupiter may be another such soliton. A swirling hurricane bigger than Earth, it has sustained itself for at least 400 years.

The self-organizing systems championed so vociferously by the physicist Ilya Prigogine were also governed by nonlinear dynamics; indeed, the self-organized motion in a simmering pot of soup turned out to be governed by dynamics very similar to the nonlinear formation of other kinds of patterns, such as the stripes of a zebra or the spots on a butterfly's wings.

But most startling of all was the nonlinear phenomenon known as chaos.

then its behavior would be too simple. It would be like a billboard where most of the bulbs just pulsed on and off like mindless strobe lights. That wasn't the kind of order Kauffman had in mind; he wanted his genetic light bulbs to organize themselves into interesting patterns analogous to waving palm trees or dancing flamingos. Besides, he knew that very sparsely connected networks were unrealistic: Jacob and Monod had already demonstrated that real genes tended to be controlled by several other genes. (Today, the number is known to be typically two to ten.)

So Kauffman started to concentrate on networks in between, where the connections were sparse, but not too sparse. To keep things simple, in fact, he looked at networks with precisely two inputs per gene. And here he began to find hints of something special. He already knew that densely connected networks were hypersensitive in the extreme: if you went in and flipped the state of any one gene from say, *on* to *off*, then you would trigger a whole avalanche of changes that would cascade back and forth through the network indefinitely. That's why densely connected networks tended to be chaotic. They could never settle down. But in his two-input networks, Kauffman discovered that flipping one gene would typically *not* produce an ever-expanding wave of change. Most often, the flipped gene would simply unflip, going back to what it was before. In fact, so long as the two different patterns of gene activation were not too different, they would tend to converge. "Things were simplifying," says Kauffman. "I could see that light bulbs tended to get into states where they got stuck on or off." In other words, the two-input networks were like a billboard where you could start the lights blinking at random, and yet they would always organize themselves into a flamingo or a champagne glass.

Order! Stealing whatever time he could from his medical courses, Kauffman filled his notebooks with more and more of his random two-input networks, analyzing the behavior of each one in detail. It was both tantalizing and frustrating work. The good news was that the two-input networks almost always seemed to stabilize very quickly. At most they would cycle over and over through a handful of different states. That's exactly what you wanted for a stable cell. The bad news was that he couldn't tell if his models had anything at all to do with real genetic regulatory networks. Real networks in real cells involved tens of thousands of genes. And yet Kauffman's pencil-and-paper networks were already getting out of hand when they contained only five or six genes. Keeping track of all the possible states and state transitions of a seven-gene network meant filling in a matrix containing 128 rows and 14 columns. Doing it for an eight-gene network would have

required a matrix twice as big as that, and so on. "The chances for making an error by hand were just awfully large," says Kauffman. "I kept looking longingly at my seven-element networks. I just couldn't stand the idea of having to do eight."

"Anyway," he says, "somewhere in my sophomore year in medical school I couldn't take it anymore. I'd been playing long enough. So I went across the street to the computer center, and I asked if someone would help me program it. They said, 'Sure, but you have to pay for it.' So I whipped out my wallet. I was ready to pay for it."

Having decided to take the plunge into computers, Kauffman vowed to go all out: he would simulate a network with 100 genes. Looking back on it now, he laughs, it was a good thing he didn't quite know what he was doing. Think of it this way. One gene by itself can have only two states: *on* and *off*. But a network of two genes can have 2×2, or four states: *on-on*, *on-off*, *off-on*, *off-off*. A network of three genes can have $2 \times 2 \times 2$, or eight states, and so on. So the number of states in a network of 100 genes is 2 multiplied by itself 100 times, which turns out to be almost exactly equal to one million trillion trillion: 1 followed by 30 zeros. That's an immense space of possibilities, says Kauffman. In principle, moreover, there was no reason why his simulated network shouldn't have just wandered around in that space at random; after all, he was deliberately wiring it up at random. And that would have meant that his idea of cell cycles was hopeless: the computer would have had to go through roughly one million trillion trillion transitions before it ever started retracing its steps. It would be a cell cycle of sorts, but vast beyond imagining. "If it takes a microsecond for the computer to go from one state to another," says Kauffman, "and if it had to keep running for something like a million trillion trillion microseconds, you'd have billions of times the history of the universe. I'd have never made it through medical school!" Indeed, the computer charges alone would have bankrupted him long before graduation.

Fortunately, however, Kauffman hadn't done that calculation at the time. So, with the aid of a very helpful programmer at the computer center, he coded up a simulated two-input network with 100 genes in it, and then blithely turned in his deck of punch cards at the front desk. The answer came back ten minutes later, printed out on wide sheets of fan-fold paper. And exactly as he had expected, it showed his network quickly settling into orderly states, with most of the genes frozen on or off and the rest cycling through a handful of configurations. These patterns certainly didn't look like flamingos or anything recognizable; if his 100-gene network had been

a Las Vegas billboard with a hundred light bulbs, then the orderly states would have looked like oscillating blobs. But they were there, and they were stable.

"I was just *unbelievably* thrilled!" says Kauffman. "And I felt then and feel now that it was rather profound. I had found something that no one would have intuited then." Instead of wandering through a space of one million trillion trillion states, his two-input network had quickly moved to an infinitesimal corner of that space and stayed there. "It settled down and oscillated through a cycle of five or six or seven or, more typically it turned out, about ten states. That's an amazing amount of order! I was just stunned."

That first simulation was only the beginning. Kauffman still had no idea of *why* sparsely connected networks were so magical. But they were, and he felt as though they had given him a whole new way of thinking about genes and embryonic development. Using that original program as a template and modifying it as needed, he ran simulations in endless variety. When and why did this orderly behavior occur, he wanted to know. And, not incidentally, how could he test his theory with real data?

Well, he thought, one obvious prediction of his model was that real genetic networks would have to be sparsely connected; densely connected networks seemed incapable of settling down into stable cycles. He didn't expect them to have precisely two inputs per gene, like his model networks. Nature is never quite that regular. But from his computer simulations and his reams of calculations, he realized that the connections only had to be sparse in a certain statistical sense. And when you looked at the data, by golly, real networks seemed to be sparse in exactly that way.

So far so good. Another test of the theory was to look at a given organism with a given set of regulatory genes, and ask how many cell types it was capable of producing. Kauffman knew he couldn't say anything specifically, of course, since he was deliberately trying to study the *typical* behavior of networks. But he could certainly look for a statistical relationship. His presumption all along had been that a cell type corresponded to one of his stable-state cycles. So he began to run bigger and bigger simulations, keeping track of how many state cycles occurred as the size of the model network increased. By the time he got up to networks of 400 to 500 genes, he had determined that the number of cycles scaled roughly as the square root of the number of genes in the network. Meanwhile, he had also been spending every spare hour in the medical school library, pouring through obscure references looking for comparable data on real organisms. And when he finally plotted it all up, there it was: the number of cell types in an organism did indeed scale roughly as the square root of the number of genes it had.

And so it went. "Goddamn it, it worked!" says Kauffman. It was the most beautiful thing he had ever experienced. By the end of his sophomore year at medical school he had run up hundreds upon hundreds of dollars in computer bills. He paid it all without a quiver.

In 1966, at the beginning of his third year of medical school, Kauffman wrote a letter to the neurophysiologist Warren McCulloch of MIT, explaining what he had done with his genetic network models and asking if McCulloch was interested.

It took a certain chutzpah to write that letter, Kauffman admits. Originally trained as an MD himself, McCulloch was one of the grand old men of neurophysiology—not to mention computer science, artificial intelligence, and the philosophy of mind. For the past two decades, he and a band of loyal followers had been working out the implications of an idea first put forward in 1943, when he and an eighteen-year-old mathematician named Walter Pitts had published a paper entitled "A Logical Calculus of the Ideas Immanent in Nervous Activity." In that paper, McCulloch and Pitts had claimed that the brain could be modeled as a network of logical operations such as *and*, *or*, *not*, and so forth. It had been a revolutionary idea at the time, to put it mildly, and had proved to be immensely influential. Not only was the McCulloch-Pitts model the first example of what would now be called a neural network, it was the first attempt to understand mental activity as a form of information processing—an insight that provided the inspiration for artificial intelligence and cognitive psychology alike. Their model was also the first indication that a network of very simple logic gates could perform exceedingly complex computations—an insight that was soon incorporated into the general theory of computing machines.

Grand old man or not, however, McCulloch seemed to be the only scientist Kauffman could share his work with. "McCulloch was the only person I knew who had done a lot of stuff with neural networks," he says. "And it was clear that genetic networks and neural networks were fundamentally the same thing."

Besides, Kauffman badly needed a little outside support by that point. Medical school was shaping up to be a decidedly mixed blessing. He was certainly getting the "facts" that he'd wanted so badly as a philosophy student at Oxford. But they weren't going down very well. "I think I chafed inside at having to take other people's word for what you're supposed to do," he says. "What one had to do in medical school was to master the facts, master

is obviously *not* random. A healthy brain produces perception, thought, and action quite coherently. Moreover, the brain is obviously not static. It refines and adapts its behavior through experience. It learns. The question is, How?

Three years earlier, in 1949, Hebb had published his answer in a book entitled *The Organization of Behavior*. His fundamental idea was to assume that the brain is constantly making subtle changes in the "synapses," the points of connection where nerve impulses make the leap from one cell to the next. This assumption was a bold move on Hebb's part, since at the time he had no evidence for it whatsoever. But having made it, he argued that these synaptic changes were in fact the basis of all learning and memory. A sensory impulse coming in from the eyes, for example, would leave its trace on the neural network by strengthening all the synapses that lay along its path. Much the same thing would happen with impulses coming in from the ears or from mental activity elsewhere in the brain itself. And as a result, said Hebb, a network that started out at random would rapidly organize itself. Experience would accumulate through a kind of positive feedback: the strong, frequently used synapses would grow stronger, while the weak, seldom-used synapses would atrophy. The favored synapses would eventually become so strong that the memories would be locked in. These memories, in turn, would tend to be widely distributed over the brain, with each one corresponding to a complex pattern of synapses involving thousands or millions of neurons. (Hebb was one of the first to describe such distributed memories as "connectionist.")

But there was more. In his lecture, Licklider went on to explain Hebb's second assumption: that the selective strengthening of the synapses would cause the brain to organize itself into "cell assemblies"—subsets of several thousand neurons in which circulating nerve impulses would reinforce themselves and continue to circulate. Hebb considered these cell assemblies to be the brain's basic building blocks of information. Each one would correspond to a tone, a flash of light, or a fragment of an idea. And yet these assemblies would *not* be physically distinct. Indeed, they would overlap, with any given neuron belonging to several of them. And because of that, activating one assembly would inevitably lead to the activation of others, so that these fundamental building blocks would quickly organize themselves into larger concepts and more complex behaviors. The cell assemblies, in short, would be the fundamental quanta of thought.

Sitting there in the audience, Holland was transfixed by all this. This wasn't just the arid stimulus/response view of psychology being pushed at the time by behaviorists such as Harvard's B. F. Skinner. Hebb was talking

about what was going on inside the mind. His connectionist theory had the richness, the perpetual surprise that Holland responded so strongly to. It felt *right*. And Holland couldn't wait to do something with it. Hebb's theory was a window onto the essence of thought, and he wanted to watch. He wanted to see cell assemblies organize themselves out of random chaos and grow. He wanted to see them interact. He wanted to see them incorporate experience and evolve. He wanted to see the emergence of the mind itself. And he wanted to see all of it happen spontaneously, without external guidance.

No sooner had Licklider finished his lecture on Hebb than Holland turned to his leader on the 701 team, Nathaniel Rochester, and said, "Well, we've got this prototype machine. Let's program a neural network simulator."

And that's exactly what they did. "He programmed one," says Holland, "and I programmed another, rather different in form. We called them the 'Conceptors.' No hubris there!"

In fact, the IBM Conceptors stand as an impressive accomplishment even forty years later, when neural network simulations have long since become a standard tool in artificial intelligence research. The basic idea would still look familiar enough. In their programs, Holland and Rochester modeled their artificial neurons as "nodes"—in effect, tiny computers that can remember certain things about their internal state. They modeled their artificial synapses as abstract connections between various nodes, with each connection having a certain "weight" corresponding to the strength of the synapse. And they modeled Hebb's learning rule by adjusting the strengths as the network gained experience. However, Holland, Rochester, and their collaborators also incorporated far more details about basic neurophysiology than most neural network simulations do today, including such factors as how fast each simulated neuron fired and how "tired" it got if it was fired too often.

Not surprisingly, they had a tough time getting it all to work. Not only were their programs among the first neural network simulations ever, they marked one of the first times a computer had been used to simulate anything (as opposed to calculating numbers or sorting data). Holland gives IBM a lot of credit for its corporate patience. He and his colleagues spent uncounted hours of computer time on their networks, and even took a trip up to Montreal to consult with Hebb himself—at company expense.

But in the end, by golly, the simulations worked. "There was a lot of emergence," says Holland, still sounding excited about it. "You could start with a uniform substrate of neurons and see the cell assemblies form." And