EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

CHAPTER 7

Mechanisms in Programs and Nature

7

Mechanisms in Programs and Nature

Universality of Behavior

In the past several chapters my main purpose has been to address the fundamental question of how simple programs behave. In this chapter my purpose is now to take what we have learned and begin applying it to the study of actual phenomena in nature.

At the outset one might have thought this would never work. For one might have assumed that any program based on simple rules would always lead to behavior that was much too simple to be relevant to most of what we see in nature. But one of the main discoveries of this book is that programs based on simple rules do not always produce simple behavior.

And indeed in the past several chapters we have seen many examples where remarkably simple rules give rise to behavior of great complexity. But to what extent is the behavior obtained from simple programs similar to behavior we see in nature?

One way to get some idea of this is just to look at pictures of natural systems and compare them with pictures of simple programs.

At the level of details there are certainly differences. But at an overall level there are striking similarities. And indeed it is quite remarkable just how often systems in nature end up showing behavior that looks almost identical to what we have seen in some simple program or another somewhere in this book.

So why might this be? It is not, I believe, any kind of coincidence, or trick of perception. And instead what I suspect is that it reflects a deep correspondence between simple programs and systems in nature.

When one looks at systems in nature, one of the striking things one notices is that even when systems have quite different underlying physical, biological or other components their overall patterns of behavior can often seem remarkably similar.

And in my study of simple programs I have seen essentially the same phenomenon: that even when programs have quite different underlying rules, their overall behavior can be remarkably similar.

So this suggests that a kind of universality exists in the types of behavior that can occur, independent of the details of underlying rules.

And the crucial point is that I believe that this universality extends not only across simple programs, but also to systems in nature. So this means that it should not matter much whether the components of a system are real molecules or idealized black and white cells; the overall behavior produced should show the same universal features.

And if this is the case, then it means that one can indeed expect to get insight into the behavior of natural systems by studying the behavior of simple programs. For it suggests that the basic mechanisms responsible for phenomena that we see in nature are somehow the same as those responsible for phenomena that we see in simple programs.

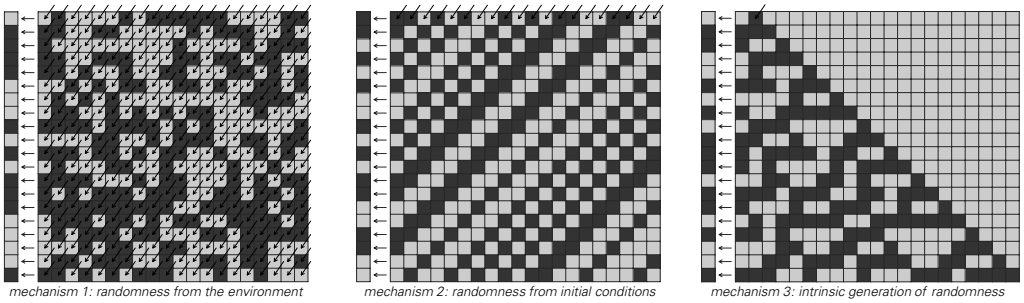
In this chapter my purpose is to discuss some of the most common phenomena that we see in nature, and to study how they correspond with phenomena that occur in simple programs.

Some of the phenomena I discuss have at least to some extent already been analyzed by traditional science. But we will find that by thinking in terms of simple programs it usually becomes possible to see the basic mechanisms at work with much greater clarity than before.

And more important, many of the phenomena that I consider—particularly those that involve significant complexity—have never been satisfactorily explained in the context of traditional science. But what we will find in this chapter is that by making use of my discoveries about simple programs a great many of these phenomena can now for the first time successfully be explained.

Three Mechanisms for Randomness

In nature one of the single most common things one sees is apparent randomness. And indeed, there are a great many different kinds of systems that all exhibit randomness. And it could be that in each case the cause of randomness is different. But from my investigations of simple programs I have come to the conclusion that one can in fact identify just three basic mechanisms for randomness, as illustrated in the pictures below.



mechanism 1: randomness from the environment *mechanism 2: randomness from initial conditions* *mechanism 3: intrinsic generation of randomness*

Three possible mechanisms that can be responsible for randomness. The diagonal arrows represent external input. In the first case, there is random input from the environment at every step. In the second case, there is random input only in the initial conditions. And in the third case, there is effectively no random input at all. Yet despite their different underlying structure, each of these mechanisms leads to randomness in the column shown at the left. The first mechanism corresponds to randomness produced by external noise, as captured in so-called stochastic models. The second mechanism is essentially the one suggested by chaos theory. The third mechanism is new, and is suggested by the results on the behavior of simple programs in this book. I will give evidence that this third mechanism is the most common one in nature.

In the first mechanism, randomness is explicitly introduced into the underlying rules for the system, so that a random color is chosen for every cell at each step.

This mechanism is the one most commonly considered in the traditional sciences. It corresponds essentially to assuming that there is a random external environment which continually affects the system one is looking at, and continually injects randomness into it.

In the second mechanism shown above, there is no such interaction with the environment. The initial conditions for the system are chosen randomly, but then the subsequent evolution of the system is assumed to follow definite rules that involve no randomness.

A crucial feature of these rules, however, is that they make the system behave in a way that depends sensitively on the details of its initial conditions. In the particular case shown, the rules are simply set up to shift every color one position to the left at each step.

And what this does is to make the sequence of colors taken on by any particular cell depend on the colors of cells progressively further and further to the right in the initial conditions. Insofar as the initial conditions are random, therefore, so also will the sequence of colors of any particular cell be correspondingly random.

In general, the rules can be more complicated than those shown in the example on the previous page. But the basic idea of this mechanism for randomness is that the randomness one sees arises from some kind of transcription of randomness that is present in the initial conditions.

The two mechanisms for randomness just discussed have one important feature in common: they both assume that the randomness one sees in any particular system must ultimately come from outside of that system. In a sense, therefore, neither of these mechanisms takes any real responsibility for explaining the origins of randomness: they both in the end just say that randomness comes from outside whatever system one happens to be looking at.

Yet for quite a few years, this rather unsatisfactory type of statement has been the best that one could make. But the discoveries about simple programs in this book finally allow new progress to be made.

The crucial point that we first saw on page 27 is that simple programs can produce apparently random behavior even when they are given no random input whatsoever. And what this means is that there is a third possible mechanism for randomness, which this time does not rely in any way on randomness already being present outside the system one is looking at.

If we had found only a few examples of programs that could generate randomness in this way, then we might think that this third mechanism was a rare and special one. But in fact over the past few chapters we have seen that practically every kind of simple program that we can construct is capable of generating such randomness.

And as a result, it is reasonable to expect that this same mechanism should also occur in many systems in nature. Indeed, as I will discuss in this chapter and the chapters that follow, I believe that this mechanism is in fact ultimately responsible for a large fraction, if not essentially all, of the randomness that we see in the natural world.

But that is not to say that the other two mechanisms are never relevant in practice. For even though they may not be able to explain how randomness is produced at the lowest level, they can still be useful in describing observations about randomness in particular systems.

And in the next few sections, I will discuss various kinds of systems where the randomness that is seen can be best described by each of the three mechanisms for randomness identified here.

Randomness from the Environment

With the first mechanism for randomness discussed in the previous section, the randomness of any particular system is taken to be the result of continual interaction between that system and randomness in its environment.

As an everyday example, we can consider a boat bobbing up and down on a rough ocean. There is nothing intrinsically random about the boat itself. But the point is that there is randomness in the continually changing ocean surface that forms the environment for the boat. And since the motion of the boat follows this ocean surface, it also seems random.

But what is the real origin of this apparent randomness? In a sense it is that there are innumerable details about an ocean that it is very difficult to know, but which can nevertheless affect the motion of the boat. Thus, for example, a particular wave that hits the boat could be the result of a nearby squall, of an undersea ridge, or perhaps even of a storm that happened the day before several hundred miles away. But since one realistically cannot keep track of all these things, the ocean will inevitably seem in many respects unpredictable and random.

This same basic effect can be even more pronounced when one looks at smaller-scale systems. A classic example is so-called Brownian

motion, in which one takes a small grain, say of pollen, puts it in a liquid, and then looks at its motion under a microscope.

What one finds is that the grain jumps around in an apparently random way. And as was suspected when this was first noticed in the 1820s, what is going on is that molecules in the liquid are continually hitting the grain and causing it to move. But even in a tiny volume of liquid there are already an immense number of molecules. And since one certainly does not even know at any given time exactly where all these molecules are, the details of their effect on the motion of the grain will inevitably seem quite random.

But to observe random Brownian motion, one needs a microscope. And one might imagine that randomness produced by any similar molecular process would also be too small to be of relevance in everyday life. But in fact such randomness is quite obvious in the operation of many kinds of electronic devices.

As an example, consider a radio receiver that is tuned to the wrong frequency or has no antenna connected. The radio receiver is built to amplify any signal that it receives. But what happens when there is no signal for it to amplify?

The answer is that the receiver produces noise. And it turns out that in most cases this noise is nothing other than a highly amplified version of microscopic processes going on inside the receiver.

In practice, such noise is usually considered a nuisance, and indeed modern digital electronics systems are typically designed to get rid of it at every stage. But since at least the 1940s, there have been various devices built for the specific purpose of generating randomness using electronic noise.

Typically these devices work by operating fairly standard electronic components in extreme conditions where there is usually no output signal, but where microscopic fluctuations can cause breakdown processes to occur which yield large output signals.

A large-scale example is a pair of metal plates with air in between. Usually no current flows across this air gap, but when the voltage between the plates is large enough, the air can break down, sparks can be generated, and spikes of current can occur. But exactly when and

where the sparks occur depends on the detailed microscopic motion of the molecules in the gas, and is therefore potentially quite random.

In an effort to obtain as much randomness as possible, actual devices that work along these lines have typically used progressively smaller components: first vacuum tubes and later semiconductors. And indeed, in a modern semiconductor diode, for example, a breakdown event can be initiated by the motion of just one electron.

But despite such sensitivity to microscopic effects, what has consistently been found in practice is that the output from such devices has significant deviations from perfect randomness.

At first, this is quite surprising. For one might think that microscopic physical processes would always produce the best possible randomness. But there are two important effects which tend to limit this randomness, or indeed any randomness that is obtained through the mechanism of interaction with the environment.

The first of these concerns the internal details of whatever device is used to sample the randomness in the environment.

Every time the device receives a piece of input, its internal state changes. But in order for successive pieces of input to be treated in an independent and uncorrelated way, the device must be in exactly the same state when it receives each piece of input. And the problem is that while practical devices may eventually relax to what is essentially the same state, they can do this only at a certain rate.

In a device that produces a spark, for example, it inevitably takes some time for the hot gas in the path of the spark to be cleared out. And if another spark is generated before this has happened, the path of the second spark will not be independent of the first.

One might think that such effects could be avoided by allowing a certain "dead time" between successive events. But in fact, as we will also see in connection with quantum mechanics, it is a rather general feature of systems that perform amplification that relaxation to a normal state can effectively occur only gradually, so that one would have to wait an infinite time for such relaxation to be absolutely complete.

But even when the device used to sample the environment does no amplification and has no relevant internal structure, one may still not see

perfect randomness. And the reason for this is that there are almost inevitably correlations even in the supposedly random environment.

In an ocean for example, the inertia of the water essentially forces there to be waves on the surface of certain sizes. And during the time that a boat is caught up in a particular one of these waves, its motion will always be quite regular; it is only when one watches the effect of a sequence of waves that one sees behavior that appears in any way random.

In a sense, though, this point just emphasizes the incomplete nature of the mechanism for randomness that we have been discussing in this section. For to know in any real way why the motion of the boat is random, we must inevitably ask more about the randomness of the ocean surface. And indeed, it is only at a fairly superficial level of description that it is useful to say that the randomness in the motion of the boat comes from interaction with an environment about which one will say nothing more than that it is random.

Chaos Theory and Randomness from Initial Conditions

At the beginning of this chapter I outlined three basic mechanisms that can lead to apparent randomness. And in the previous section I discussed the first of these mechanisms—based on the idea that the evolution of a system is continually affected by randomness from its environment.

But to get randomness in a particular system it turns out that there is no need for continual interaction between the system and an external random environment. And in the second mechanism for randomness discussed at the beginning of this chapter, no explicit randomness is inserted during the evolution of a system. But there is still randomness in the initial conditions, and the point is that as the system evolves, it samples more and more of this randomness, and as a result produces behavior that is correspondingly random.

As a rather simple example one can think of a car driving along a bumpy road. Unlike waves on an ocean, all the bumps on the road are already present when the car starts driving, and as a result, one can consider these bumps to be part of the initial conditions for the system. But the point is that as time goes on, the car samples more and more of

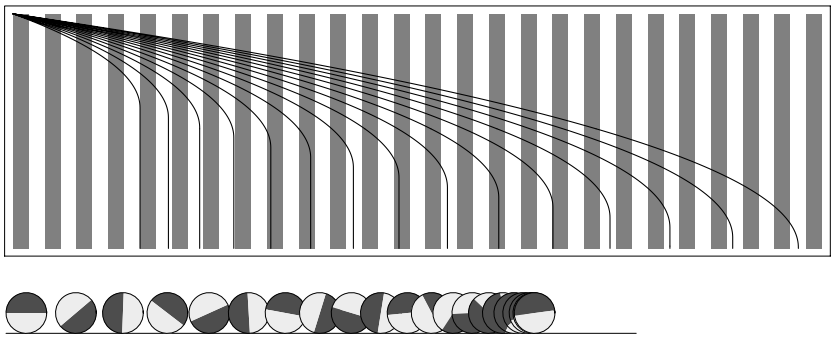
the bumps, and if there is randomness in these bumps it leads to corresponding randomness in the motion of the car.

A somewhat similar example is a ball rolled along a rough surface. A question such as where the ball comes to rest will depend on the pattern of bumps on the surface. But now another feature of the initial conditions is also important: the initial speed of the ball.

And somewhat surprisingly there is already in practice some apparent randomness in the behavior of such a system even when there are no significant bumps on the surface. Indeed, games of chance based on rolling dice, tossing coins and so on all rely on just such randomness.

As a simple example, consider a ball that has one hemisphere white and the other black. One can roll this ball like a die, and then look to see which color is on top when the ball comes to rest. And if one does this in practice, what one will typically find is that the outcome seems quite random. But where does this randomness come from?

The answer is that it comes from randomness in the initial speed with which the ball is rolled. The picture below shows the motion of a ball with a sequence of different initial speeds. And what one sees is that it takes only a small change in the initial speed to make the ball come to rest in a completely different orientation.



A plot of the position of a ball rolled with various initial speeds. Time goes down the page. The ball starts on the left, with an initial speed given by the initial slope of the curve. The ball slows down as a result of friction, and eventually stops. The ball is half white and half black, and the stripes in the picture indicate which color is on top when the ball is at a particular position. The divergence of the curves in the picture indicate the sensitivity of the motion to the exact initial speed of the ball. Small changes in this speed are seen to make the ball stop with a different color on top. It is such sensitivity to randomness in the initial conditions that makes processes such as rolling dice or tossing coins yield seemingly random output.

The point then is that a human rolling the ball will typically not be able to control this speed with sufficient accuracy to determine whether black or white will end up on top. And indeed on successive trials there will usually be sufficiently large random variations in the initial speed that the outcomes will seem completely random.

Coin tossing, wheels of fortune, roulette wheels, and similar generators of randomness all work in essentially the same way. And in each case the basic mechanism that leads to the randomness we see is a sensitive dependence on randomness that is present in the typical initial conditions that are provided.

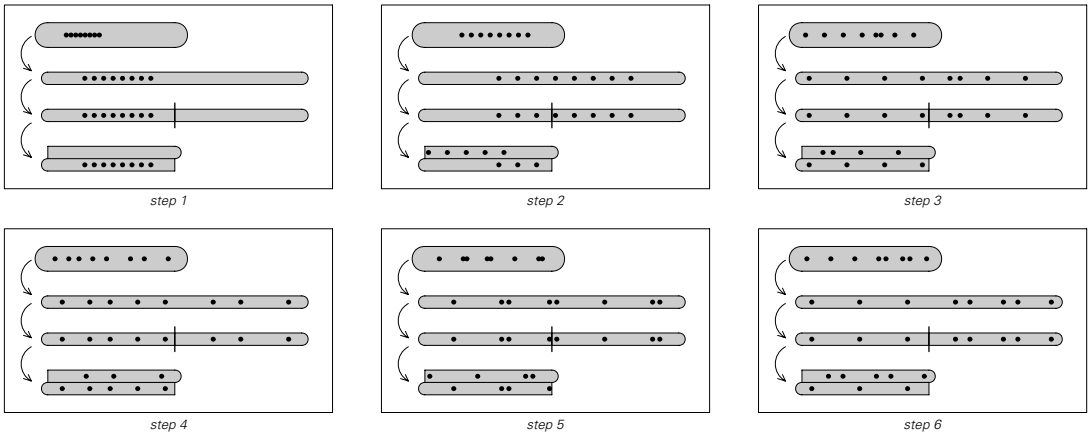
Without randomness in the initial conditions, however, there is no randomness in the output from these systems. And indeed it is quite feasible to build precise machines for tossing coins, rolling balls and so on that always produce a definite outcome with no randomness at all.

But the discovery which launched what has become known as chaos theory is that at least in principle there can be systems whose sensitivity to their initial conditions is so great that no machine with fixed tolerances can ever be expected to yield repeatable results.

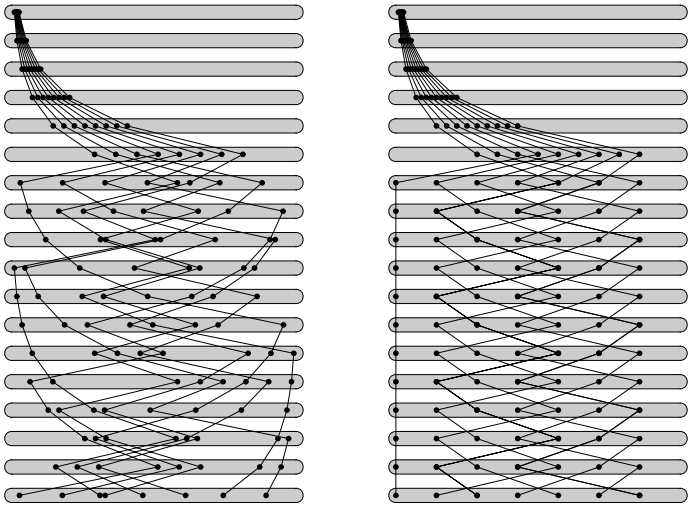
A classic example is an idealized version of the kneading process which is used for instance to make noodles or taffy. The basic idea is to take a lump of dough-like material, and repeatedly to stretch this material to twice its original length, cut it in two, then stack the pieces on top of each other. The picture at the top of the facing page shows a few steps in this process. And the important point to notice is that every time the material is stretched, the distance between neighboring points is doubled.

The result of this is that any change in the initial position of a point will be amplified by a factor of two at each step. And while a particular machine may be able to control the initial position of a point to a certain accuracy, such repeated amplification will eventually lead to sensitivity to still smaller changes.

But what does this actually mean for the motion of a point in the material? The bottom pictures on the facing page show what happens to two sets of points that start very close together. The most obvious effect is that these points diverge rapidly on successive steps. But after a while, they reach the edge of the material and cannot diverge any



A kneading process similar to ones used to make noodles or taffy, which exhibits very sensitive dependence on initial conditions. In the first part of each step, the material is stretched to twice its original length. Then it is cut in two, and the two halves are stacked on top of each other. The picture demonstrates that dots which are initially close together rapidly separate. (A more realistic kneading process would fold material rather than cutting it, but the same sensitive dependence on initial conditions would occur.)



Two examples of what can happen when the kneading process above is applied to nearby collections of points. In both cases the points initially diverge exponentially, as implied by chaos theory. But after a while they reach the edge of the material, and although in the first case they then show quite random behavior, in the second case they instead just show simple repetitive behavior. What differs between the two cases is the detailed digit sequences of the positions of the points: in the first case these digit sequences are quite random, while in the second case they have a simple repetitive form.

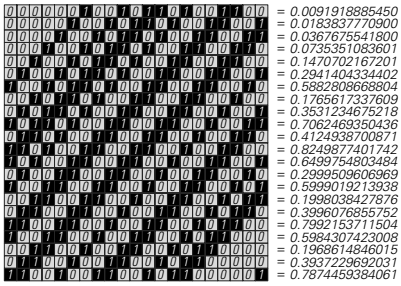
further. And then in the first case, the subsequent motion looks quite random. But in the second case it is fairly regular. So why is this?

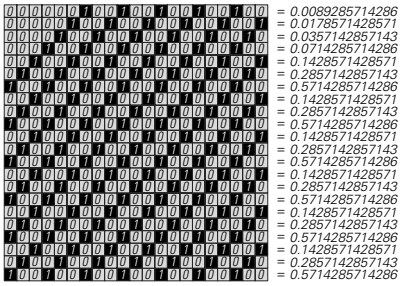
A little analysis shows what is going on. The basic idea is to represent the position of each point at each step as a number, say x, which runs from 0 to 1. When the material is stretched, the number is

doubled. And when the material is cut and stacked, the effect on the number is then to extract its fractional part.

But it turns out that this process is exactly the same as the one we discussed on page 153 in the chapter on systems based on numbers.

And what we found there was that it is crucial to think not in terms of the sizes of the numbers x, but rather in terms of their digit sequences represented in base 2. And in fact, in terms of such digit sequences, the kneading process consists simply in shifting all digits one place to the left at each step, as shown in the pictures below.





The digit sequences of positions of points on successive steps in the two examples of kneading processes at the bottom of the previous page. At each step these digit sequences are shifted one place to the left. So if the initial digit sequence is random, as in the first example, then the subsequent behavior will also be correspondingly random. But if the initial digit sequence is simple, as in the second example, then the behavior will be correspondingly simple. In general, a point at position x on a particular step will move to position *FractionalPart[2 x]* on the next step.

The way digit sequences work, digits further to the right in a number always make smaller contributions to its overall size. And as a result, one might think that digits which lie far to the right in the initial conditions would never be important. But what the pictures above show is that these digits will always be shifted to the left, so that eventually they will in fact be important. As time goes on, therefore, what is effectively happening is that the system is sampling digits further and further to the right in the initial conditions.

And in a sense this is not unlike what happens in the example of a car driving along a bumpy road discussed at the beginning of this section. Indeed in many ways the only real difference is that instead of

being able to see a sequence of explicit bumps in the road, the initial conditions for the position of a point in the kneading process are encoded in a more abstract form as a sequence of digits.

But the crucial point is that the behavior we see will only ever be as random as the sequence of digits in the initial conditions. And in the first case on the facing page, it so happens that the sequence of digits for each of the initial points shown is indeed quite random, so the behavior we see is correspondingly random. But in the second case, the sequence of digits is regular, and so the behavior is correspondingly regular.

Sensitive dependence on initial conditions thus does not in and of itself imply that a system will behave in a random way. Indeed, all it does is to cause digits which make an arbitrarily small contribution to the size of numbers in the initial conditions eventually to have a significant effect. But in order for the behavior of the system to be random, it is necessary in addition that the sequence of digits be random. And indeed, the whole idea of the mechanism for randomness in this section is precisely that any randomness we see must come from randomness in the initial conditions for the system we are looking at.

It is then a separate question why there should be randomness in these initial conditions. And ultimately this question can only be answered by going outside of the system one is looking at, and studying whatever it was that set up its initial conditions.

Accounts of chaos theory in recent years have, however, often introduced confusion about this point. For what has happened is that from an implicit assumption made in the mathematics of chaos theory, the conclusion has been drawn that random digit sequences should be almost inevitable among the numbers that occur in practice.

The basis for this is the traditional mathematical idealization that the only relevant attribute of any number is its size. And as discussed on page 152, what this idealization suggests is that all numbers which are sufficiently close in size should somehow be equally common. And indeed if this were true, then it would imply that typical initial conditions would inevitably involve random digit sequences.

But there is no particular reason to believe that an idealization which happens to be convenient for mathematical analysis should

apply in the natural world. And indeed to assume that it does is effectively just to ignore the fundamental question of where randomness in nature comes from.

But beyond even such matters of principle, there are serious practical problems with the idea of getting randomness from initial conditions, at least in the case of the kneading process discussed above.

The issue is that the description of the kneading process that we have used ignores certain obvious physical realities. Most important among these is that any material one works with will presumably be made of atoms. And as a result, the notion of being able to make arbitrarily small changes in the position of a point is unrealistic.

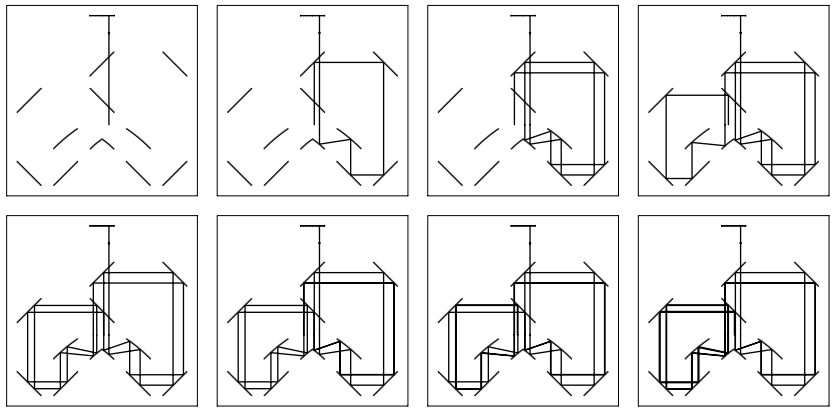
One might think that atoms would always be so small that their size would in practice be irrelevant. But the whole point is that the kneading process continually amplifies distances. And indeed after just thirty steps, the description of the kneading process given above would imply that two points initially only one atom apart would end up nearly a meter apart.

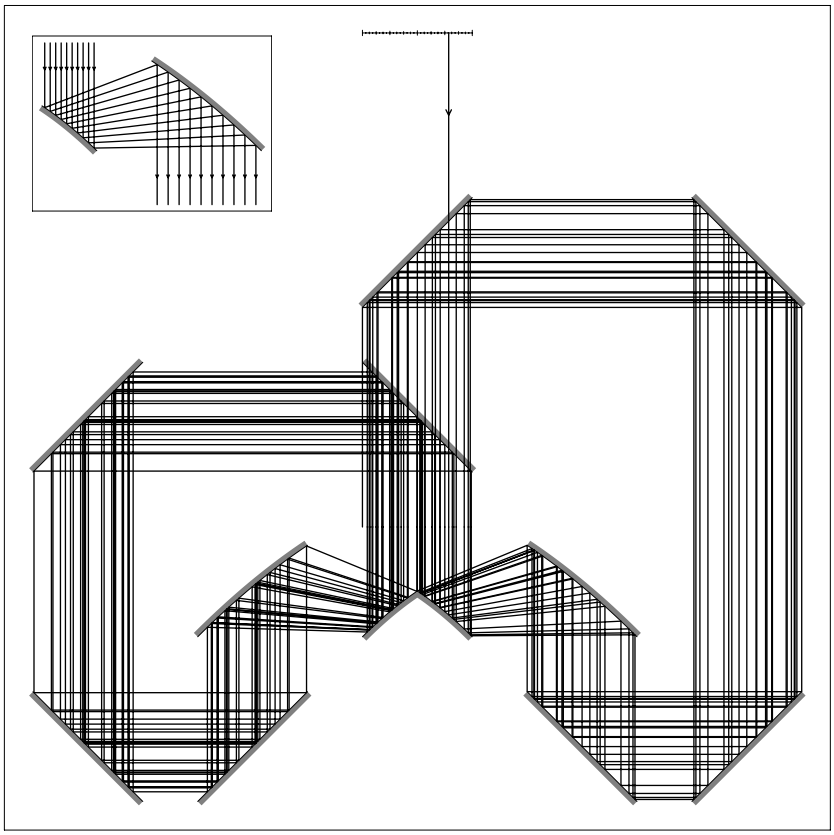
Yet long before this would ever happen in practice other effects not accounted for in our simple description of the kneading process would inevitably also become important. And often such effects will tend to introduce new randomness from the environment. So the idea that randomness comes purely from initial conditions can be realistic only for a fairly small number of steps; randomness which is seen after that must therefore typically be attributed to other mechanisms.

One might think that the kneading process we have been discussing is just a bad example, and that in other cases, randomness from initial conditions would be more significant.

The picture on the facing page shows a system in which a beam of light repeatedly bounces off a sequence of mirrors. The system is set up so that every time the light goes around, its position is modified in exactly the same way as the position of a point in the kneading process. And just as in the kneading process, there is very sensitive dependence on the details of the initial conditions, and the behavior that is seen reflects the digit sequence of these initial conditions.

But once again, in any practical implementation, the light would go around only a few tens of times before being affected by microscopic





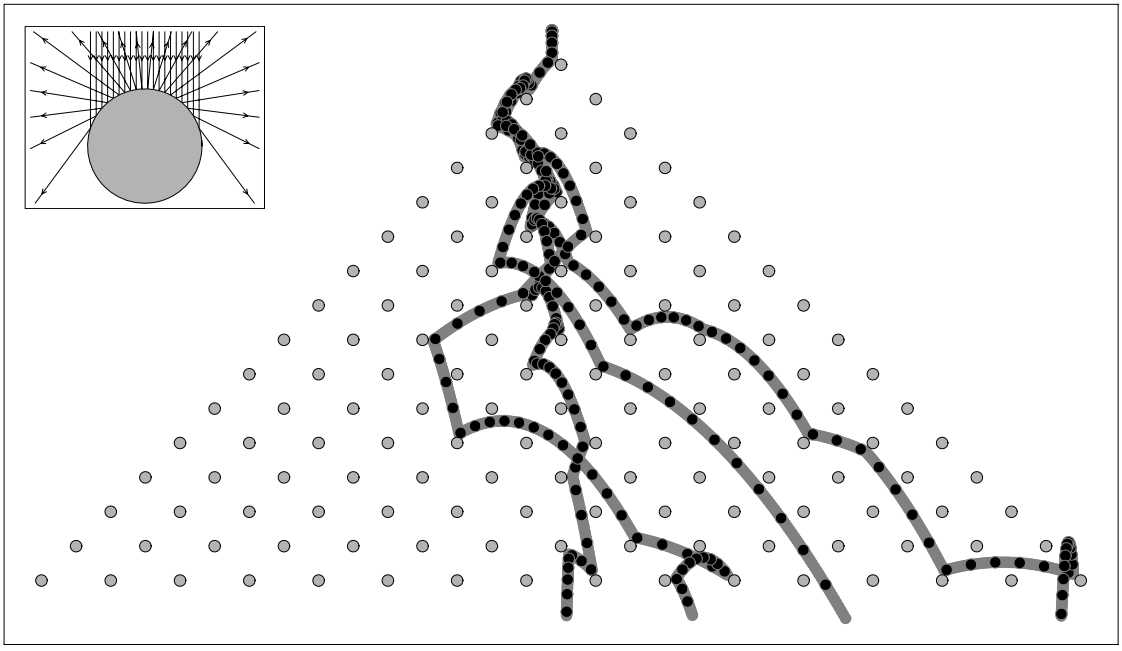
An arrangement of mirrors set up to exhibit randomness arising from sensitive dependence on initial conditions. The initial condition for the system is specified by the position of the incoming light ray in the gray region at the top of each picture. Whether the light ray goes to the left or to the right at each step is then determined by successive digits in the base 2 representation for the number that gives the initial condition. The heart of the system is the "amplifier" shown on the left which uses a pair of parabolic mirrors to double the displacement of each incoming ray. The initial condition used here is $\pi/4$, which has digit sequence 0.1100100100001111111.

perturbations in the mirrors and by other phenomena that are not accounted for in the simple description we have given.

At the heart of the system shown on the previous page is a slightly complicated arrangement of parabolic mirrors. But it turns out that almost any convex reflector will lead to the divergence of trajectories necessary to get sensitive dependence on initial conditions.

Indeed, the simple pegboard shown below exhibits the same phenomenon, with balls dropped at even infinitesimally different initial positions eventually following very different trajectories.

The details of these trajectories cannot be deduced quite as directly as before from the digit sequences of initial positions, but



Paths followed by four idealized balls dropped from initial positions differing by one part in a thousand into an array of identical circular pegs. The balls are taken to fall under gravity, and to bounce elastically whenever they hit a peg. As illustrated in the inset, small differences in direction are amplified—roughly doubling—at each bounce, with the result that after a few bounces the trajectories of the three balls are quite different. In a physical version of the system with balls of the same actual size as on this page perturbations from the environment will inevitably be amplified to have a significant effect on the trajectories after roughly the number of bounces shown. Versions of the system illustrated here—particularly with smaller peg spacings—are sometimes known as Galton or quincunx boards, and have been used since the late 1800s to demonstrate principles of probability theory. If balls are assumed to fall randomly on each side of each peg then with a large number of balls the final positions will approximate a binomial distribution.

exactly the same phenomenon of successively sampling less and less significant digits still occurs. And once again, at least for a while, any randomness in the motion of the ball can be attributed to randomness in this initial digit sequence.

But after at most ten or so collisions, many other effects, mostly associated with continual interaction with the environment, will always in practice become important, so that any subsequent randomness cannot solely be attributed to initial conditions.

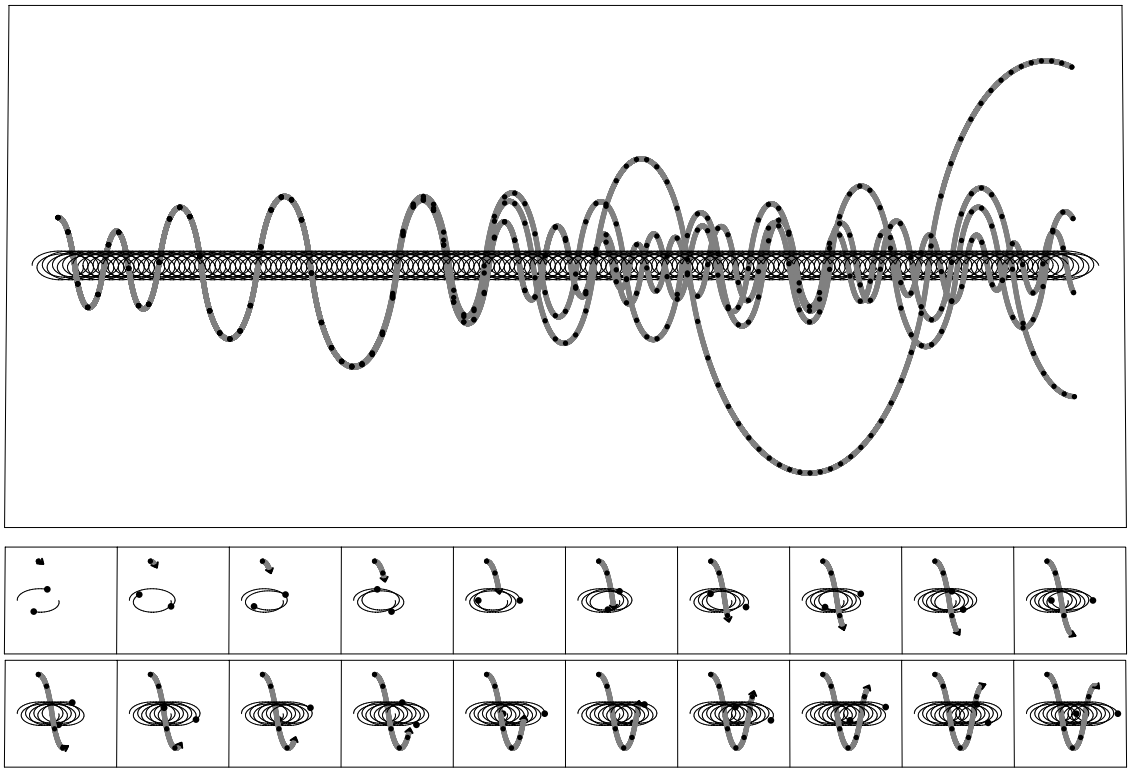
And indeed in any system, the amount of time over which the details of initial conditions can ever be considered the dominant source of randomness will inevitably be limited by the level of separation that exists between the large-scale features that one observes and small-scale features that one cannot readily control.

So in what kinds of systems do the largest such separations occur? The answer tends to be systems in astronomy. And as it turns out, the so-called three-body problem in astronomy was the very first place where sensitive dependence on initial conditions was extensively studied.

The three-body problem consists in determining the motion of three bodies—such as the Earth, Sun and Moon—that interact through gravitational attraction. With just two bodies, it has been known for nearly four hundred years that the orbits that occur are simple ellipses or hyperbolas. But with three bodies, the motion can be much more complicated, and—as was shown at the end of the 1800s—can be sensitively dependent on the initial conditions that are given.

The pictures on the next page show a particular case of the three-body problem, in which there are two large masses in a simple elliptical orbit, together with an infinitesimally small mass moving up and down through the plane of this orbit. And what the pictures demonstrate is that even if the initial position of this mass is changed by just one part in a hundred million, then within 50 revolutions of the large masses the trajectory of the small mass will end up being almost completely different.

So what happens in practice with planets and other bodies in our solar system? Observations suggest that at least on human timescales most of their motion is quite regular. And in fact this regularity was in



An example of the three-body problem, in which an idealized planet moves up and down through the plane of two equal-mass idealized stars in a perfect elliptical orbit. The trajectories obtained with four possible initial positions for the planet—differing by 10^{-8}—are shown. The pictures are made assuming the system to be in uniform motion from left to right. Successive black dots indicate where the planets are on each revolution of the stars. The main picture shows what happens over the course of 100 revolutions. The planet is assumed to be of negligible mass relative to the stars, and to start with zero vertical velocity at exactly an equal distance between the stars. The divergence of trajectories with slightly different initial vertical positions indicates sensitive dependence on initial conditions.

the past taken as one of the key pieces of evidence for the idea that simple laws of nature could exist.

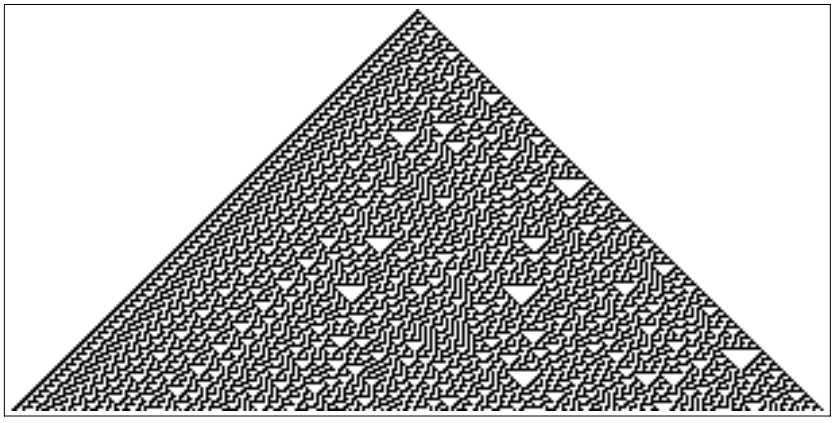
But calculations imply that sensitive dependence on initial conditions should ultimately occur even in our solar system. Needless to say, we do not have the option of explicitly setting up different initial conditions. But if we could watch the solar system for a few million years, then there should be significant randomness that could be attributed to sensitive dependence on the digit sequences of initial conditions—and whose presence in the past may explain some observed present-day features of our solar system.

The Intrinsic Generation of Randomness

In the past two sections, we have studied two possible mechanisms that can lead to observed randomness. But as we have discussed, neither of these in any real sense themselves generate randomness. Instead, what they essentially do is just to take random input that comes from outside, and transfer it to whatever system one is looking at.

One of the important results of this book, however, is that there is also a third possible mechanism for randomness, in which no random input from outside is needed, and in which randomness is instead generated intrinsically inside the systems one is looking at.

The picture below shows the rule 30 cellular automaton in which I first identified this mechanism for randomness. The basic rule for the system is very simple. And the initial condition is also very simple.



The rule 30 cellular automaton from page 27 that was the first example I found of intrinsic randomness generation. There is no random input to this system, yet its behavior seems in many respects random. I suspect that this is how much of the randomness that we see in nature arises.

Yet despite the lack of anything that can reasonably be considered random input, the evolution of the system nevertheless intrinsically yields behavior which seems in many respects random.

As we have discussed before, traditional intuition makes it hard to believe that such complexity could arise from such a simple

underlying process. But the past several chapters have demonstrated that this is not only possible, but actually quite common.

Yet looking at the cellular automaton on the previous page there are clearly at least some regularities in the pattern it produces—like the diagonal stripes on the left. But if, say, one specifically picks out the color of the center cell on successive steps, then what one gets seems like a completely random sequence.

But just how random is this sequence really?

For our purposes here the most relevant point is that so far as one can tell the sequence is at least as random as sequences one gets from any of the phenomena in nature that we typically consider random.

When one says that something seems random, what one usually means in practice is that one cannot see any regularities in it. So when we say that a particular phenomenon in nature seems random, what we mean is that none of our standard methods of analysis have succeeded in finding regularities in it. To assess the randomness of a sequence produced by something like a cellular automaton, therefore, what we must do is to apply to it the same methods of analysis as we do to natural systems.

As I will discuss in Chapter 10, some of these methods have been well codified in standard mathematics and statistics, while others are effectively implicit in our processes of visual and other perception. But the remarkable fact is that none of these methods seem to reveal any real regularities whatsoever in the rule 30 cellular automaton sequence. And thus, so far as one can tell, this sequence is at least as random as anything we see in nature.

But is it truly random?

Over the past century or so, a variety of definitions of true randomness have been proposed. And according to most of these definitions, the sequence is indeed truly random. But there are a certain class of definitions which do not consider it truly random.

For these definitions are based on the notion of classifying as truly random only sequences which can never be generated by any simple procedure whatsoever. Yet starting with a simple initial condition and then applying a simple cellular automaton rule constitutes a simple

procedure. And as a result, the center column of rule 30 cannot be considered truly random according to such definitions.

But while definitions of this type have a certain conceptual appeal, they are not likely to be useful in discussions of randomness in nature. For as we will see later in this book, it is almost certainly impossible for any natural process ever to generate a sequence which is guaranteed to be truly random according to such definitions.

For our purposes more useful definitions tend to concentrate not so much on whether there exists in principle a simple way to generate a particular sequence, but rather on whether such a way can realistically be recognized by applying various kinds of analysis to the sequence. And as discussed above, there is good evidence that the center column of rule 30 is indeed random according to all reasonable definitions of this kind.

So whether or not one chooses to say that the sequence is truly random, it is, as far as one can tell, at least random for all practical purposes. And in fact sequences closely related to it have been used very successfully as sources of randomness in practical computing.

For many years, most kinds of computer systems and languages have had facilities for generating what they usually call random numbers. And in *Mathematica*—ever since it was first released—Random[Integer] has generated 0's and 1's using exactly the rule 30 cellular automaton.

The way this works is that every time Random[Integer] is called, another step in the cellular automaton evolution is performed, and the value of the cell in the center is returned. But one difference from the picture two pages ago is that for practical reasons the pattern is not allowed to grow wider and wider forever. Instead, it is wrapped around in a region that is a few hundred cells wide.

One consequence of this, as discussed on page 259, is that the sequence of 0's and 1's that is generated must then eventually repeat. But even with the fastest foreseeable computers, the actual period of repetition will typically be more than a billion billion times the age of the universe.

Another issue is that if one always ran the cellular automaton from page 315 with the particular initial condition shown there, then one would always get exactly the same sequence of 0's and 1's. But by using different initial conditions one can get completely different

sequences. And in practice if the initial conditions are not explicitly specified, what *Mathematica* does, for example, is to use as an initial condition a representation of various features of the exact state of the computer system at the time when Random was first called.

The rule 30 cellular automaton provides a particularly clear and good example of intrinsic randomness generation. But in previous chapters we have seen many other examples of systems that also intrinsically produce apparent randomness. And it turns out that one of these is related to the method used since the late 1940s for generating random numbers in almost all practical computer systems.

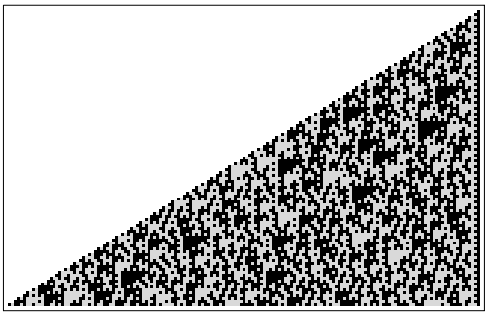
The pictures on the facing page show what happens if one successively multiplies a number by various constant factors, and then looks at the digit sequences of the numbers that result. As we first saw on page 119, the patterns of digits obtained in this way seem quite random. And the idea of so-called linear congruential random number generators is precisely to make use of this randomness.

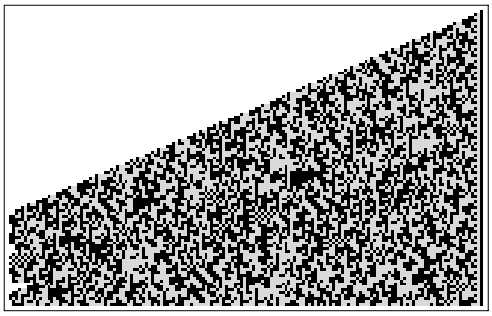
For practical reasons, such generators typically keep only, say, the rightmost 31 digits in the numbers at each step. Yet even with this restriction, the sequences generated are random enough that at least until recently they were almost universally what was used as a source of randomness in practical computing.

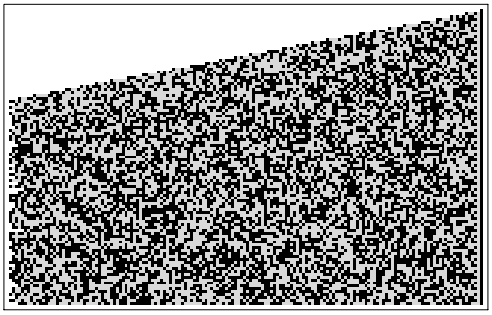
So in a sense linear congruential generators are another example of the general phenomenon of intrinsic randomness generation. But it turns out that in some respects they are rather unusual and misleading.

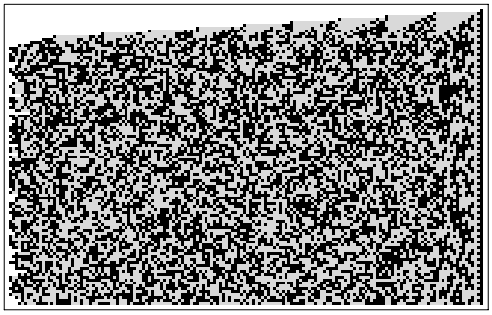
Keeping only a limited number of digits at each step makes it inevitable that the sequences produced will eventually repeat. And one of the reasons for the popularity of linear congruential generators is that with fairly straightforward mathematical analysis it is possible to tell exactly what multiplication factors will maximize this repetition period.

It has then often been assumed that having maximal repetition period will somehow imply maximum randomness in all aspects of the sequence one gets. But in practice over the years, one after another linear congruential generator that has been constructed to have maximal repetition period has turned out to exhibit very substantial deviations from perfect randomness.


multiplier 3


multiplier 5

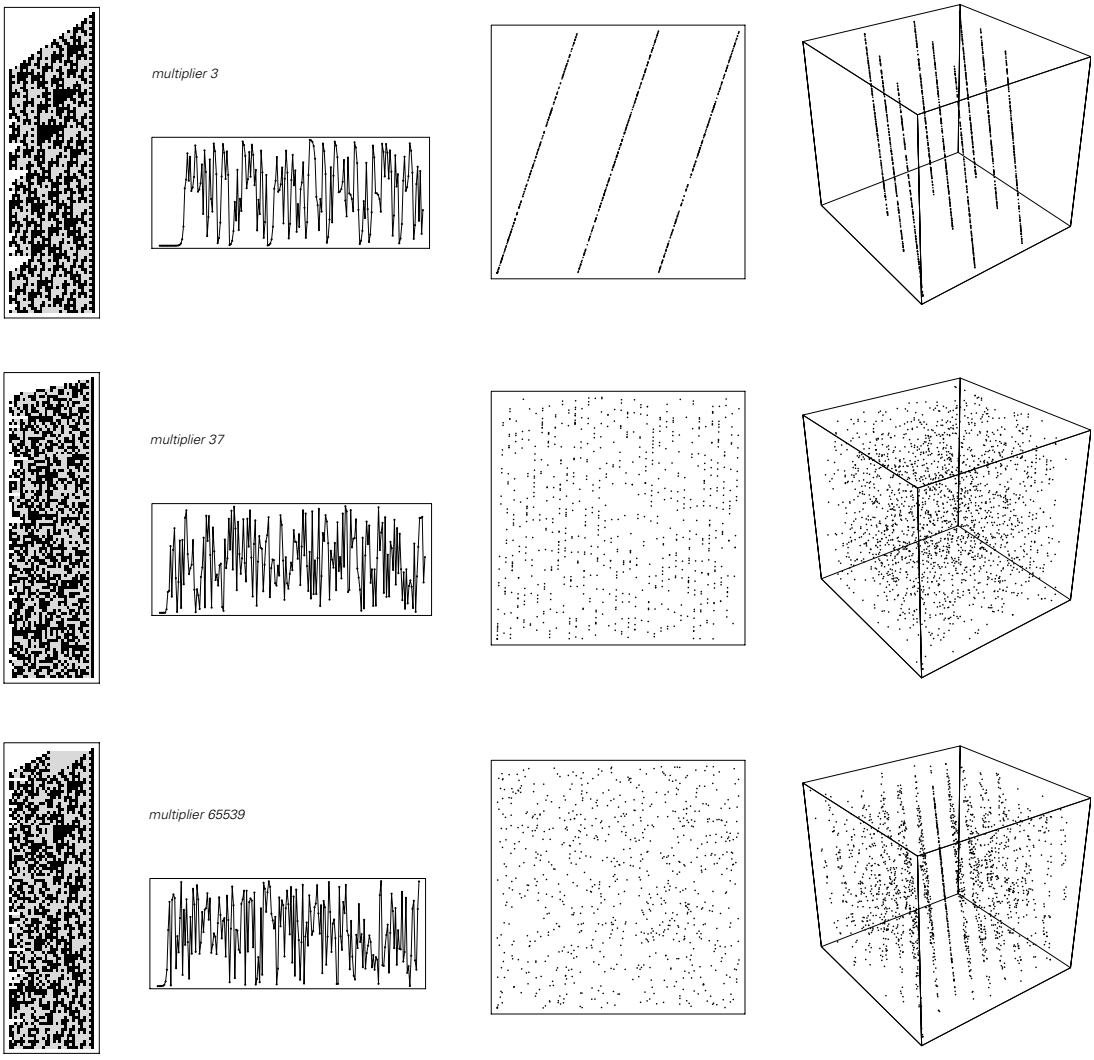

multiplier 37


multiplier 65539

Patterns of digits in base 2 produced by starting with the number 1 and then repeatedly multiplying by various fixed constants. In all cases, the complete pattern has a triangular form, but except in the first case, it is truncated on the left here. The mathematical structure of these systems is nevertheless such that digits further to the left do not affect those shown: at each step the number obtained is effectively reduced modulo 2^n, where n is the width of the picture.

A typical kind of failure, illustrated in the pictures on the next page, is that points with coordinates determined by successive numbers from the generator turn out to be distributed in an embarrassingly regular way. At first, such failures might suggest that more complicated schemes must be needed if one is to get good randomness. And indeed with this thought in mind all sorts of elaborate combinations of linear congruential and other generators have been proposed. But although some aspects of the behavior of such systems can be made quite random, deviations from perfect randomness are still often found.

And seeing this one might conclude that it must be essentially impossible to produce good randomness with any kind of system that has reasonably simple rules. But the rule 30 cellular automaton that we discussed above demonstrates that in fact this is absolutely not the case.



Examples of three so-called linear congruential random number generators. In each case they start with the number 1, then successively multiply by the specified multiplier, keeping only the rightmost 31 digits in the base 2 representation of the number obtained at each step. A version of the case with multiplier 3 was already shown on page 120. Multiplier 65539 was used as the random number generator on many computer systems, starting with mainframes in the 1960s. The last two pictures in each row above give the distribution of points whose coordinates in two and three dimensions are obtained by taking successive numbers from the linear congruential generator. If the output from the generator was perfectly random, then in each case these points would be uniformly distributed. But as the pictures demonstrate, stripes are visible in either two or three dimensions, or both.

Indeed, the rules for this cellular automaton are in some respects much simpler than for even a rather basic linear congruential generator. Yet the sequences it produces seem perfectly random, and do not suffer from any of the problems that are typically found in linear congruential generators.

So why do linear congruential generators not produce better randomness? Ironically, the basic reason is also the reason for their popularity. The point is that unlike the rule 30 cellular automaton that we discussed above, linear congruential generators are readily amenable to detailed mathematical analysis. And as a result, it is possible for example to guarantee that a particular generator will indeed have a maximal repetition period.

Almost inevitably, however, having such a maximal period implies a certain regularity. And in fact, as we shall see later in this book, the very possibility of any detailed mathematical analysis tends to imply the presence of at least some deviations from perfect randomness.

But if one is not constrained by the need for such analysis, then as we saw in the cellular automaton example above, remarkably simple rules can successfully generate highly random behavior.

And indeed the existence of such simple rules is crucial in making it plausible that the general mechanism of intrinsic randomness generations can be widespread in nature. For if the only way for intrinsic randomness generation to occur was through very complicated sets of rules, then one would expect that this mechanism would be seen in practice only in a few very special cases.

But the fact that simple cellular automaton rules are sufficient to give rise to intrinsic randomness generation suggests that in reality it is rather easy for this mechanism to occur. And as a result, one can expect that the mechanism will be found often in nature.

So how does the occurrence of this mechanism compare to the previous two mechanisms for randomness that we have discussed?

The basic answer, I believe, is that whenever a large amount of randomness is produced in a short time, intrinsic randomness generation is overwhelmingly likely to be the mechanism responsible.

We saw in the previous section that random details of the initial conditions for a system can lead to a certain amount of randomness in

the behavior of a system. But as we discussed, there is in most practical situations a limit on the lengths of sequences whose randomness can realistically be attributed to such a mechanism. With intrinsic randomness generation, however, there is no such limit: in the cellular automaton above, for example, all one need do to get a longer random sequence is to run the cellular automaton for more steps.

But it is also possible to get long random sequences by continual interaction with a random external environment, as in the first mechanism for randomness discussed in this chapter.

The issue with this mechanism, however, is that it can take a long time to get a given amount of good-quality randomness from it. And the point is that in most cases, intrinsic randomness generation can produce similar randomness in a much shorter time.

Indeed, in general, intrinsic randomness generation tends to be much more efficient than getting randomness from the environment. The basic reason is that intrinsic randomness generation in a sense puts all the components in a system to work in producing new randomness, while getting randomness from the environment does not.

Thus, for example, in the rule 30 cellular automaton discussed above, every cell in effect actively contributes to the randomness we see. But in a system that just amplifies randomness from the environment, none of the components inside the system itself ever contribute any new randomness at all. Indeed, ironically enough, the more components that are involved in the process of amplification, the slower it will typically be to get each new piece of random output. For as we discussed two sections ago, each component in a sense adds what one can consider to be more inertia to the amplification process.

But with a larger number of components it becomes progressively easier for randomness to be generated through intrinsic randomness generation. And indeed unless the underlying rules for the system somehow explicitly prevent it, it turns out in the end that intrinsic randomness generation will almost inevitably occur—often producing so much randomness that it completely swamps any randomness that might be produced from either of the other two mechanisms.

Yet having said this, one can ask how one can tell in an actual experiment on some particular system in nature to what extent intrinsic randomness generation is really the mechanism responsible for whatever seemingly random behavior one observed.

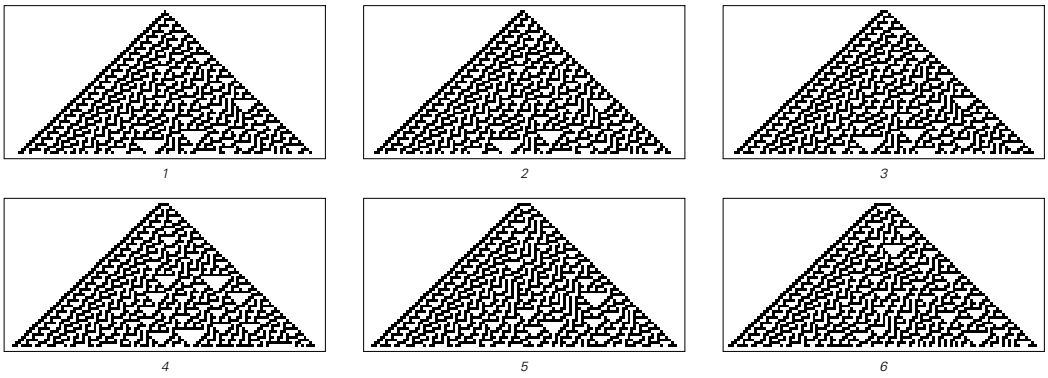
The clearest sign is a somewhat unexpected phenomenon: that details of the random behavior can be repeatable from one run of the experiment to another. It is not surprising that general features of the behavior will be the same. But what is remarkable is that if intrinsic randomness generation is the mechanism at work, then the precise details of the behavior can also be repeatable.

In the mechanism where randomness comes from continual interaction with the environment, no repeatability can be expected. For every time the experiment is run, the state of the environment will be different, and so the behavior one sees will also be correspondingly different. And similarly, in the mechanism where randomness comes from the details of initial conditions, there will again be little, if any, repeatability. For the details of the initial conditions are typically affected by the environment of the system, and cannot realistically be kept the same from one run to another.

But the point is that with the mechanism of intrinsic randomness generation, there is no dependence on the environment. And as a result, so long as the setup of the system one is looking at remains the same, the behavior it produces will be exactly the same. Thus for example, however many times one runs a rule 30 cellular automaton, starting with a single black cell, the behavior one gets will always be exactly the same. And so for example the sequence of colors of the center cell, while seemingly random, will also be exactly the same.

But how easy is it to disturb this sequence? If one makes a fairly drastic perturbation, such as changing the colors of cells all the way from white to black, then the sequence will indeed often change, as illustrated in the pictures at the top of the next page.

But with less drastic perturbations, the sequence can be quite robust. As an example, one can consider allowing each cell to be not just black or white, but any shade of gray, as in the continuous cellular automata we discussed on page 155. And in such systems, one can



The effect of changing the number of initial black cells in the rule 30 cellular automaton shown above. With only 2 or 3 black cells, the sequence in the center of the pattern does not change. But as soon as more black cells are added, it does change.

investigate what happens if at every step one randomly perturbs the gray level of each cell by a small amount.

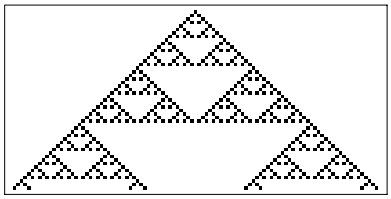
The pictures on the facing page show results for perturbations of various sizes. What one sees is that when the perturbations are sufficiently large, the sequence of colors of the center cell does indeed change. But the crucial point is that for perturbations below a certain critical size, the sequence always remains essentially unchanged.

Even though small perturbations are continually being made, the evolution of the system causes these perturbations to be damped out, and produces behavior that is in practice indistinguishable from what would be seen if there were no perturbations.

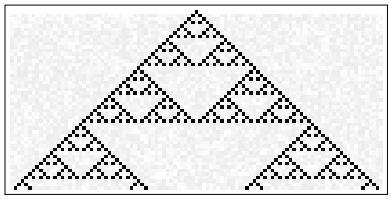
The question of what size of perturbations can be tolerated without significant effect depends on the details of the underlying rules. And as the pictures suggest, rules which yield more complex behavior tend to be able to tolerate only smaller sizes of perturbations. But the crucial point is that even when the behavior involves intrinsic randomness generation, perturbations of at least some size can still be tolerated.

And the reason this is important is that in any real experiment, there are inevitably perturbations on the system one is looking at.

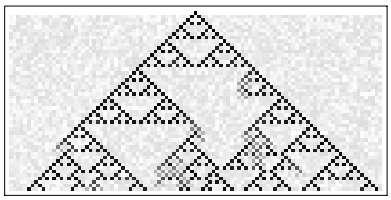
With more care in setting up the experiment, a higher degree of isolation from the environment can usually be achieved. But it is never possible to eliminate absolutely all interaction with the environment.



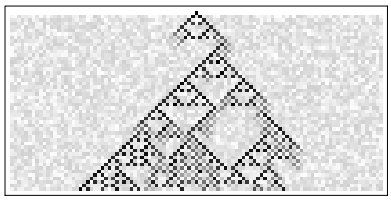
0% perturbations



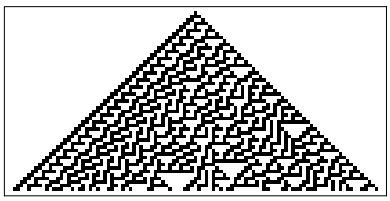
5% perturbations



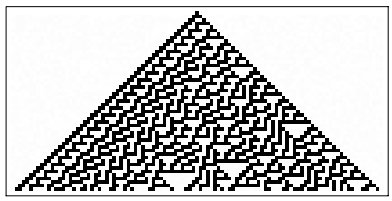
10% perturbations



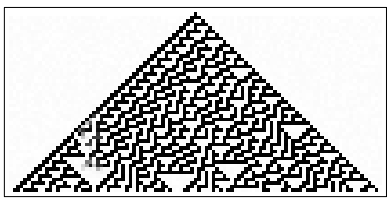
15% perturbations



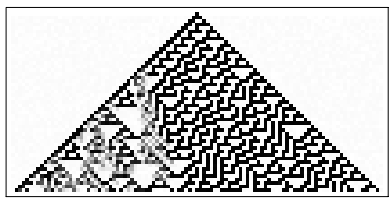
0% perturbations



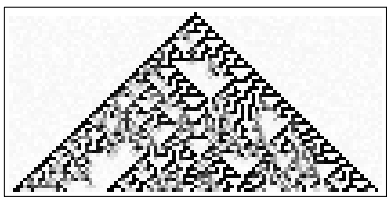
0.5% perturbations



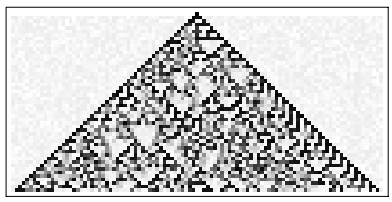
0.8% perturbations



1% perturbations

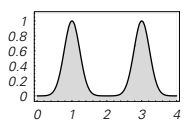


2% perturbations



5% perturbations

The effects of various levels of external randomness on the behavior of continuous cellular automata with generalizations of rules 90 and 30. The value of each cell can be any gray level between 0 and 1. For the generalization of rule 90, the values of the left and right cells are added together, and the value of the cell on the next step is then found by applying the continuous generalization of the modulo 2 function shown at the right. For the generalization of rule 30, a similar scheme based on an algebraic representation of the rule is used. In both cases, every value at each step is also perturbed by a random amount up to the percentage indicated for each picture.



And as a result, the system one is looking at will be subjected to at least some level of random perturbations from the environment.

But what the pictures on the previous page demonstrate is that when such perturbations are small enough, they will have essentially no effect. And what this means is that when intrinsic randomness generation is the dominant mechanism it is indeed realistic to expect at least some level of repeatability in the random behavior one sees in real experiments.

So has such repeatability actually been seen in practice?

Unfortunately there is so far very little good information on this point, since without the idea of intrinsic randomness generation there was never any reason to look for such repeatability when behavior that seemed random was observed in an experiment.

But scattered around the scientific literature—in various corners of physics, chemistry, biology and elsewhere—I have managed to find at least some cases where multiple runs of the same carefully controlled experiment are reported, and in which there are clear hints of repeatability even in behavior that looks quite random.

If one goes beyond pure numerical data of the kind traditionally collected in scientific experiments, and instead looks for example at the visual appearance of systems, then sometimes the phenomenon of repeatability becomes more obvious. Indeed, for example, as I will discuss in Chapter 8, different members of the same biological species often have many detailed visual similarities—even in features that on their own seem complex and apparently quite random.

And when there are, for example, two symmetrical sides to a particular system, it is often possible to compare the visual patterns produced on each side, and see what similarities exist. And as various examples in Chapter 8 demonstrate, across a whole range of physical, biological and other systems there can indeed be remarkable similarities.

So in all of these cases the randomness one sees cannot reasonably be attributed to randomness that is introduced from the environment—either continually or through initial conditions. And instead, there is no choice but to conclude that the randomness must in fact come from the mechanism of intrinsic randomness generation that I have discovered in simple programs, and discussed in this section.

The Phenomenon of Continuity

Many systems that we encounter in nature have behavior that seems in some way smooth or continuous. Yet cellular automata and most of the other programs that we have discussed involve only discrete elements. So how can such systems ever reproduce what we see in nature?

The crucial point is that even though the individual components in a system may be discrete, the average behavior that is obtained by looking at a large number of these components may still appear to be smooth and continuous. And indeed, there are many familiar systems in nature where exactly this happens.

Thus, for example, air and water seem like continuous fluids, even though we know that at a microscopic level they are both in fact made up of discrete molecules. And in a similar way, sand flows much like a continuous fluid, even though we can easily see that it is actually made up of discrete grains. So what is the basic mechanism that allows systems with discrete components to produce behavior that seems smooth and continuous?

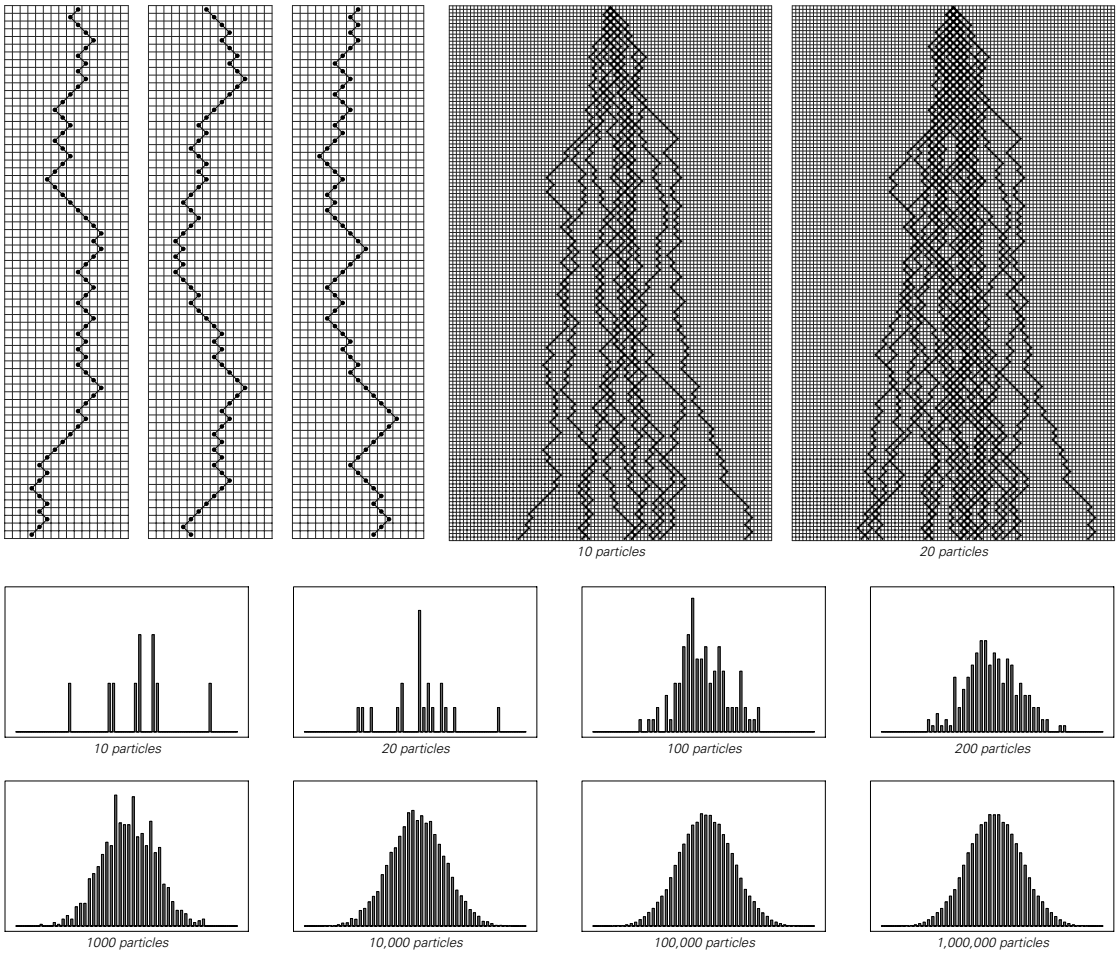
Most often, the key ingredient is randomness.

If there is no randomness, then the overall forms that one sees tend to reflect the discreteness of the underlying components. Thus, for example, the faceted shape of a crystal reflects the regular microscopic arrangement of discrete atoms in the crystal.

But when randomness is present, such microscopic details often get averaged out, so that in the end no trace of discreteness is left, and the results appear to be smooth and continuous. The next page shows a classic example of this phenomenon, based on so-called random walks.

Each random walk is made by taking a discrete particle, and then at each step randomly moving the particle one position to the left or right. If one starts off with several particles, then at any particular time, each particle will be at a definite discrete position. But what happens if one looks not at the position of each individual particle, but rather at the overall distribution of all particles?

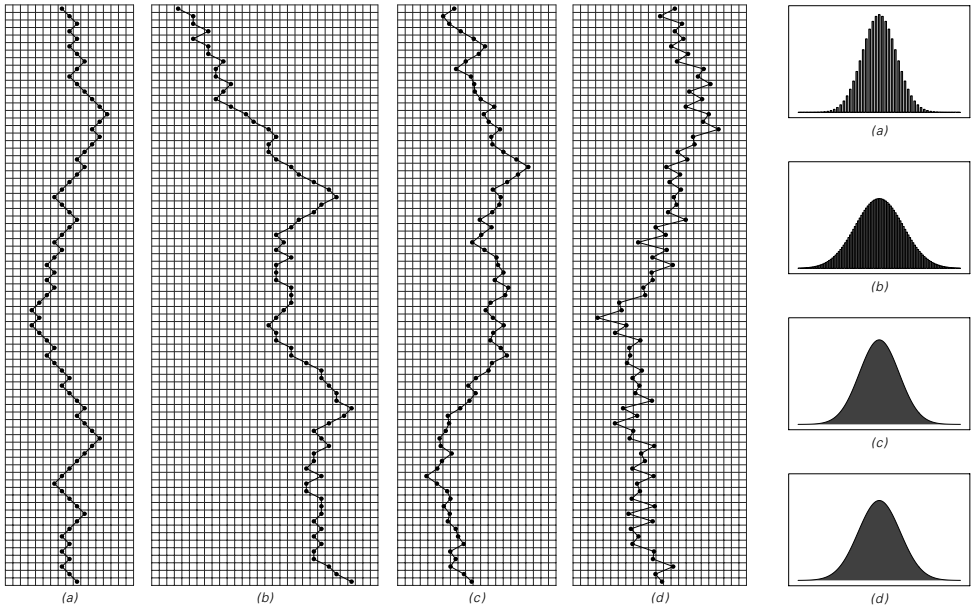
The answer, as illustrated on the next page, is that if there are enough particles, then the distribution one sees takes on a smooth and



The distribution of positions by reached particles that follow random walks. The top left shows three individual examples of random walks, in which each particle randomly moves one position to the left or right. Even though the individual particles are discrete, the pictures show that when a large number of particles are considered, the overall behavior obtained seems smooth and continuous.

continuous form, and shows no trace of the underlying discreteness of the system; the randomness has in a sense successfully washed out essentially all the microscopic details of the system.

The pictures at the top of the facing page show what happens if one uses several different underlying rules for the motion of each particle. And what one sees is that despite differences at a microscopic level, the overall distribution obtained in each case has exactly the same continuous form.

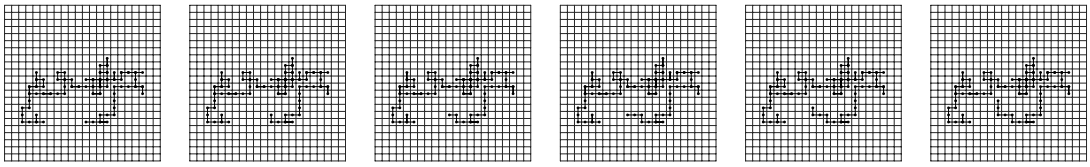


A demonstration of the fact that for a wide range of underlying rules for each step in a random walk, the overall distribution obtained always has the same continuous form. In case (a), each particle moves just one position to the left or right at each step. In case (b), it can move between 0, 1 or 2 positions, while in case (c) it can move any distance between 0 and 1 at each step. Finally, in case (d), on alternate steps the particle moves either always to the right or always to the left.

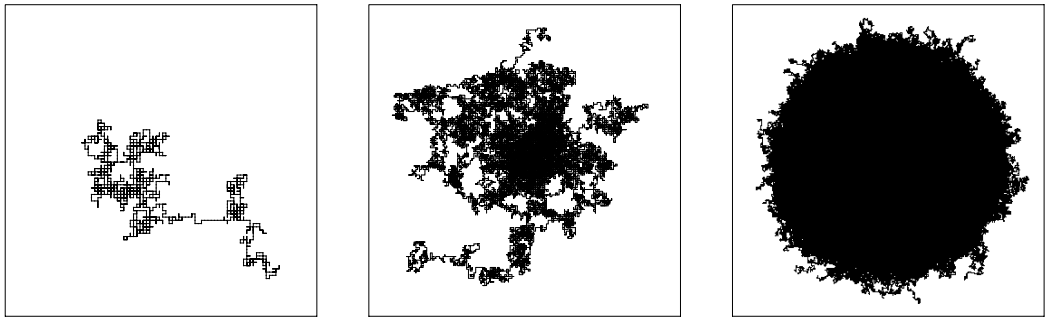
Indeed, in the particular case of systems such as random walks, the Central Limit Theorem suggested over two centuries ago ensures that for a very wide range of underlying microscopic rules, the same continuous so-called Gaussian distribution will always be obtained.

This kind of independence of microscopic details has many important consequences. The pictures on the next page show, for example, what happens if one looks at two-dimensional random walks on square and hexagonal lattices.

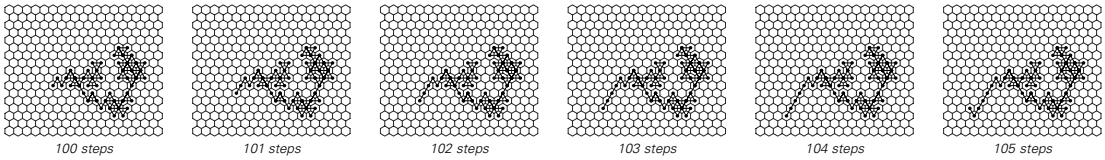
One might expect that the different underlying forms of these lattices would lead to different shapes in overall distributions. But the remarkable fact illustrated on the next page is that when enough particles are considered, one gets in the end distributions that have a purely circular shape that shows no trace of the different discrete structures of the underlying lattices.



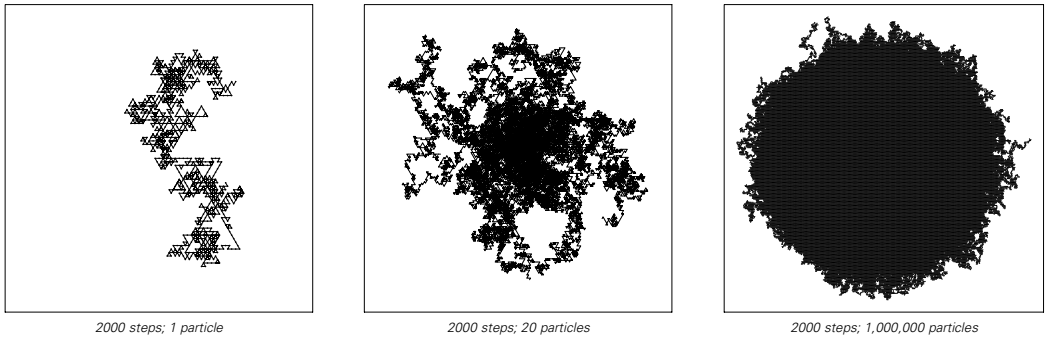
100 steps 101 steps 102 steps 103 steps 104 steps 105 steps



2000 steps; 1 particle 2000 steps; 20 particles 2000 steps; 1,000,000 particles



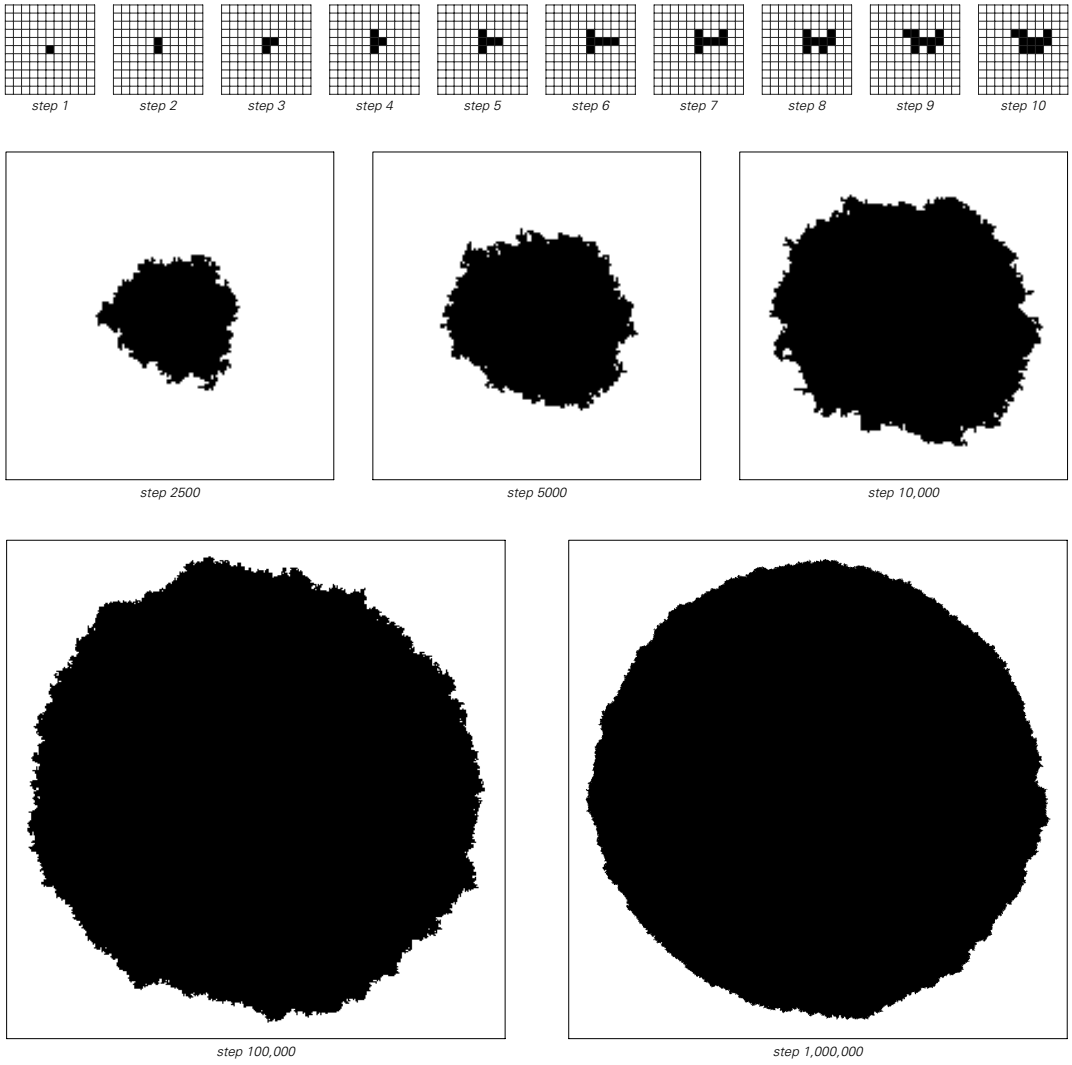
100 steps 101 steps 102 steps 103 steps 104 steps 105 steps



2000 steps; 1 particle 2000 steps; 20 particles 2000 steps; 1,000,000 particles

Examples of random walks on square and hexagonal lattices. Despite the different underlying lattices the average of sufficiently many particles yields ultimately circular behavior in both cases—as implied by the Central Limit Theorem.

Beyond random walks, there are many other systems based on discrete components in which randomness at a microscopic level also leads to continuous behavior on a large scale. The picture below shows as one example what happens in a simple aggregation model.

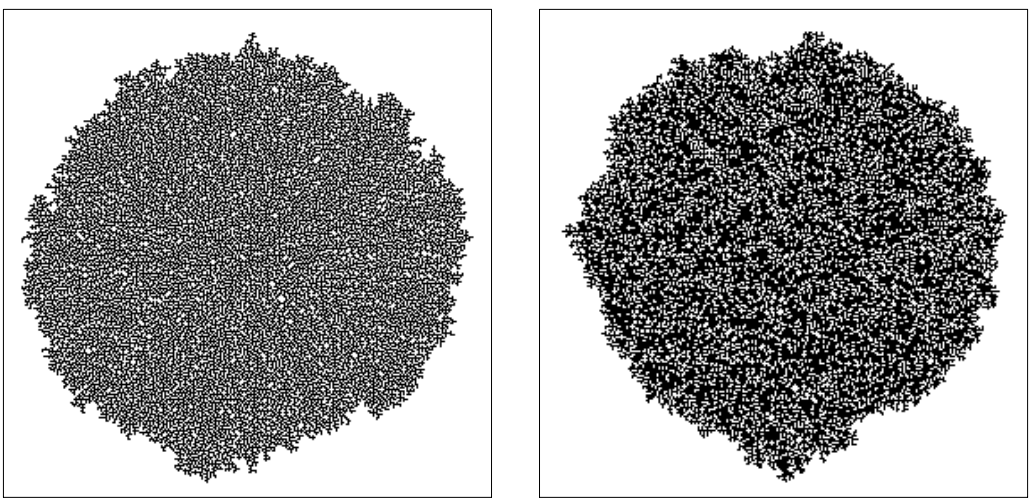


Behavior of a simple aggregation model, in which a single new black cell is added at each step at a randomly chosen position adjacent to the existing cluster of black cells. The system is a version of the so-called Eden model. The shape obtained is ultimately an almost perfect circle.

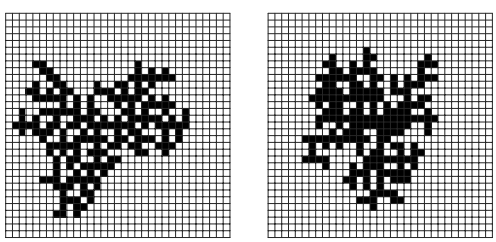
The idea of this model is to build up a cluster of black cells by adding just one new cell at each step. The position of this cell is chosen entirely at random, with the only constraint being that it should be adjacent to an existing cell in the cluster.

At early stages, clusters that are grown in this way look quite irregular. But after a few thousand steps, a smooth overall roughly circular shape begins to emerge. Unlike for the case of random walks, there is as yet no known way to make a rigorous mathematical analysis of this process. But just as for random walks, it appears once again that the details of the underlying rules for the system do not have much effect on the main features of the behavior that is seen.

The pictures below, for example, show generalizations of the aggregation model in which new cells are added only at positions that have certain numbers of existing neighbors. And despite such changes



(a) (b)



(a) (b)

Patterns produced by generalized aggregation models in which a new cell is added only if (a) it would have only one immediate neighbor (out of four), or (b) it would have either one or four neighbors. The pictures above show step 30,000, while those on the right show step 200. Despite the difference in underlying rules, the same basic overall shape of pattern is eventually produced.

in underlying rules, the overall shapes of the clusters produced remain very much the same.

In all these examples, however, the randomness that is involved comes from the same basic mechanism: it is explicitly inserted from outside at each step in the evolution of the system.

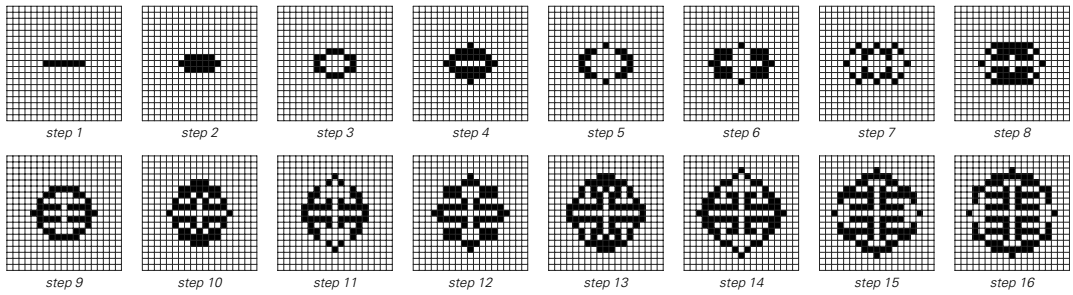
But it turns out that all that really seems to matter is that randomness is present: the mechanism through which it arises appears to be largely irrelevant. And in particular what this means is that randomness which comes from the mechanism of intrinsic randomness generation discussed in the previous section is able to make systems with discrete components behave in seemingly continuous ways.

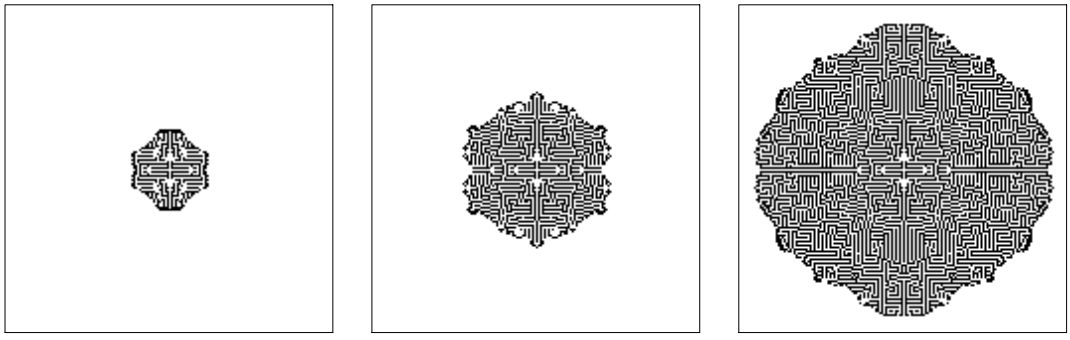
The picture on the next page shows a two-dimensional cellular automaton where this happens. There is no randomness in the rules or the initial conditions for this system. But through the mechanism of intrinsic randomness generation, the behavior of the system exhibits considerable randomness. And this randomness turns out to lead to an overall pattern of growth that yields the same basic kind of smooth roughly circular form as in the aggregation model.

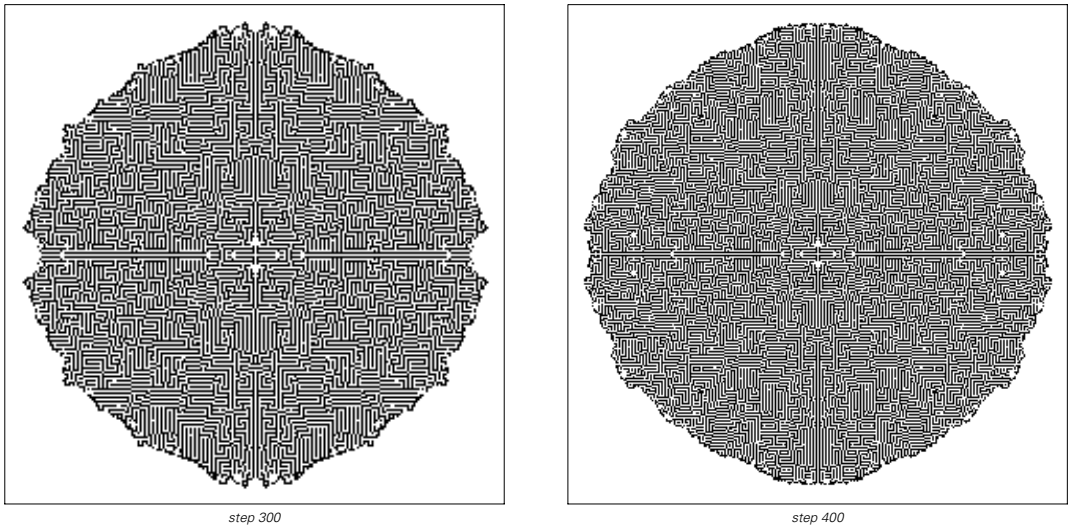
Having seen this, one might then wonder whether in fact any system that involves randomness will ultimately produce smooth overall patterns of growth. The answer is definitely no. In discussing two-dimensional cellular automata in Chapter 5, for example, we saw many examples where randomness occurs, but where the overall forms of growth that are produced have a complicated structure with no particular smoothness or continuity.

As a rough guide, it seems that continuous patterns of growth are possible only when the rate at which small-scale random changes occur is substantially greater than the overall rate of growth. For in a sense it is only then that there is enough time for randomness to average out the effects of the underlying discrete structure.

And indeed this same issue also exists for processes other than growth. In general the point is that continuous behavior can arise in systems with discrete components only when there are features that evolve slowly relative to the rate of small-scale random changes.



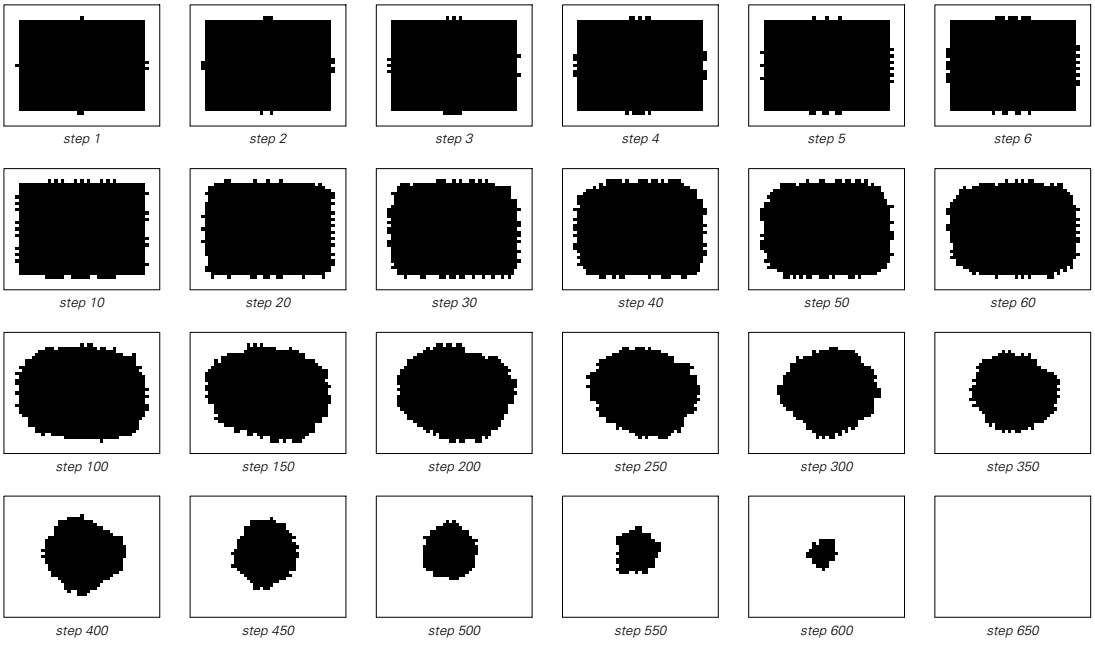




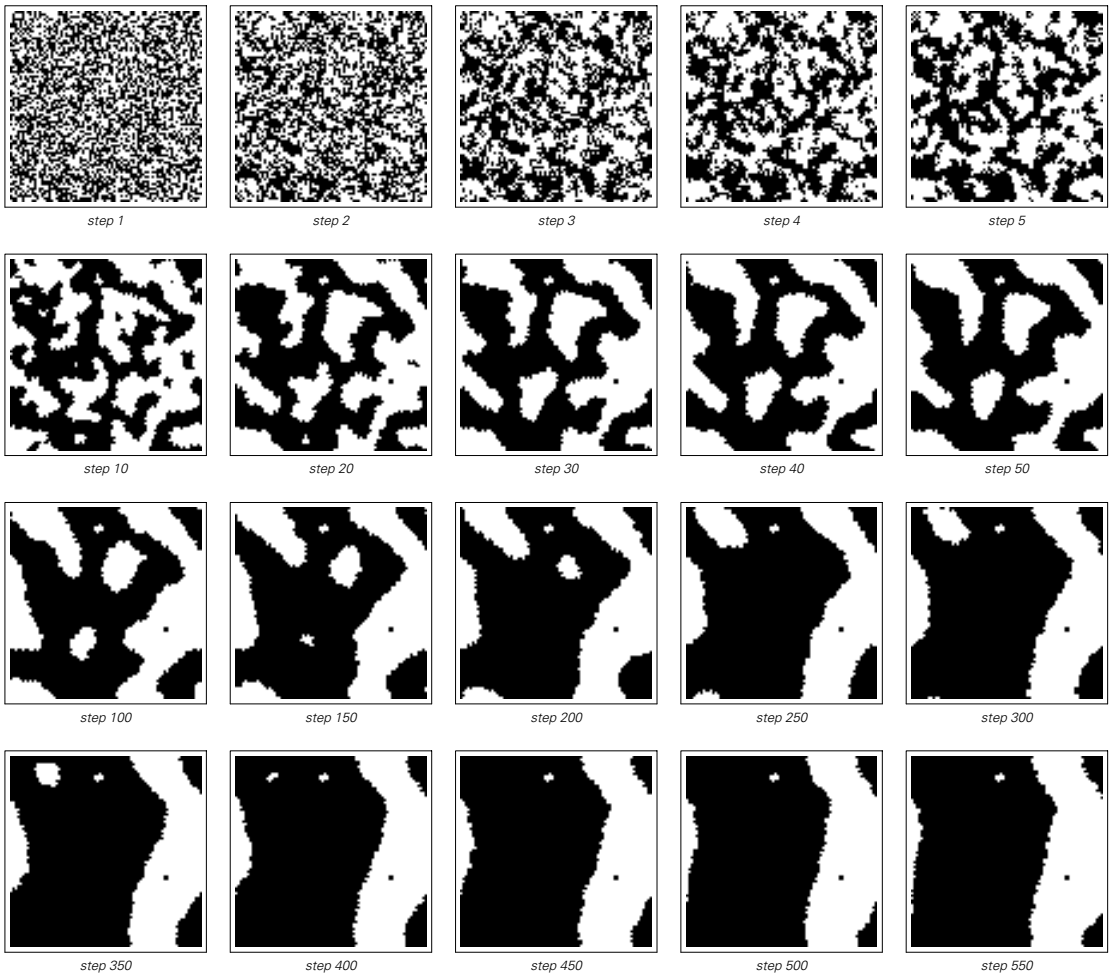
A two-dimensional cellular automaton first shown on page 178 with the rule that if out of the eight neighbors (including diagonals) around a given cell, there are exactly three black cells, then the cell itself becomes black on the next step. If the cell has 1, 2 or 4 black neighbors, then it stays the same color as before, and if it has 5 or more black neighbors, then it becomes white on the next step. (Outer totalistic code 746.) This simple rule produces randomness through the mechanism of intrinsic randomness generation, and this randomness in turn leads to a pattern of growth that takes on an increasingly smooth more-or-less circular form.

The pictures on the next page show an example where this happens. The detailed pattern of black and white cells in these pictures changes at every step. But the point is that the large domains of black and white that form have boundaries which move only rather slowly. And at an overall level these boundaries then behave in a way that looks quite smooth and continuous.

It is still true, however, that at a small scale the boundaries consist of discrete cells. But as the picture below shows, the detailed configuration of these cells changes rapidly in a seemingly random way. And just as in the other systems we have discussed, what then emerges on average from all these small-scale random changes is overall behavior that again seems in many ways smooth and continuous.



The behavior of an individual domain of black cells in the cellular automaton shown on the next page. The boundary of the domain exhibits seemingly random fluctuations. But at an overall level, the behavior that is produced seems in many respects quite smooth and continuous. The domain effectively behaves as if it has a surface tension, so that it first evolves to a roughly circular shape, then shrinks eventually to nothing. The main black rectangle is initially 39 × 29 cells in size.



Behavior of a two-dimensional cellular automaton starting from a random initial condition. At each step, each cell looks at the total number of black cells in the 9-cell neighborhood consisting of the cell itself and the 8 cells adjacent to it (including diagonals). If this total is less than 4, then the cell becomes white on the next step, while if the total is greater than 6, it becomes black. If the total is exactly 5, then the cell becomes white, and if the total is exactly 4, then it becomes black. (The rule has totalistic code 976.) The pictures show that on a large scale, the rule leads to regions of black and white whose boundaries behave in a seemingly smooth and continuous way. Note that each picture is 80 cells across, and is effectively wrapped around so that the left neighbor of the leftmost cell is the rightmost cell, and so on.

Origins of Discreteness

In the previous section we saw that even though a system may on a small scale consist of discrete components, it is still possible for the system overall to exhibit behavior that seems smooth and continuous. And as we have discussed before, the vast majority of traditional mathematical models have in fact been based on just such continuity.

But when one looks at actual systems in nature, it turns out that one often sees discrete behavior—so that, for example, the coat of a zebra has discrete black and white stripes, not continuous shades of gray. And in fact many systems that exhibit complex behavior show at least some level of overall discreteness.

So what does this mean for continuous models? In the previous section we found that discrete models could yield continuous behavior. And what we will find in this section is that the reverse is also true: continuous models can sometimes yield behavior that appears discrete.

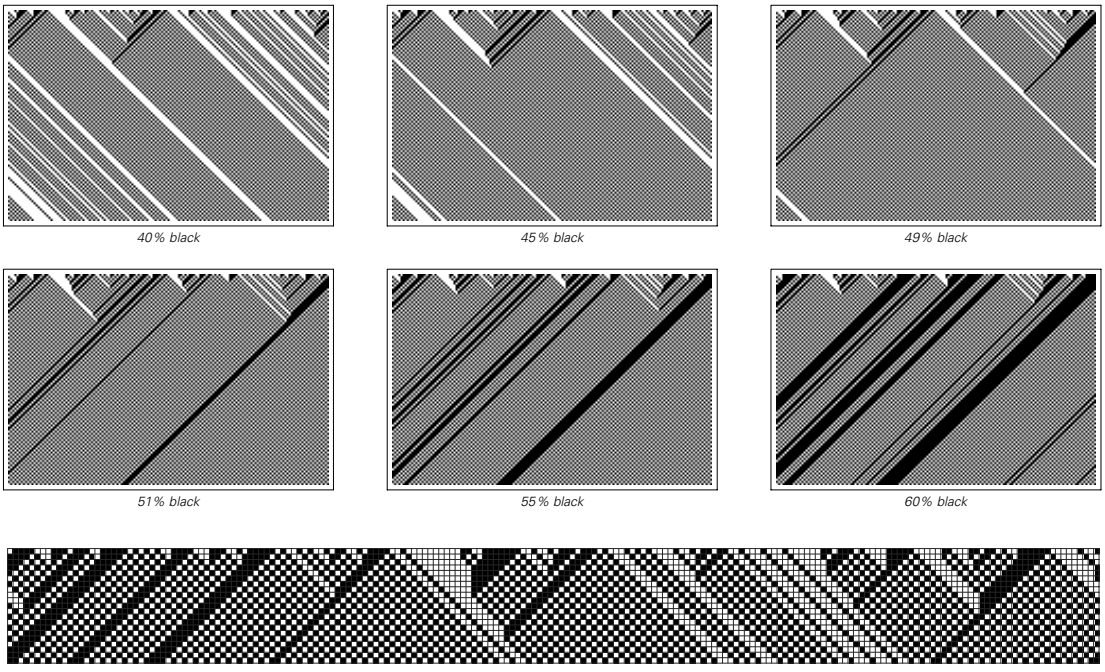
Needless to say, if one wants to study phenomena that are based on discreteness, it usually makes more sense to start with a model that is fundamentally discrete. But in making contact with existing scientific models and results, it is useful to see how discrete behavior can emerge from continuous processes.

The boiling of water provides a classic example. If one takes some water and continuously increases its temperature, then for a while nothing much happens. But when the temperature reaches 100°C, a discrete transition occurs, and all the water evaporates into steam.

It turns out that there are many kinds of systems in which continuous changes can lead to such discrete transitions.

The pictures at the top of the next page show a simple example based on a one-dimensional cellular automaton. The idea is to make continuous changes in the initial density of black cells, and then to see what effect these have on the overall behavior of the system.

One might think that if the changes one makes are always continuous, then effects would be correspondingly continuous. But the pictures on the next page demonstrate that this is not so.

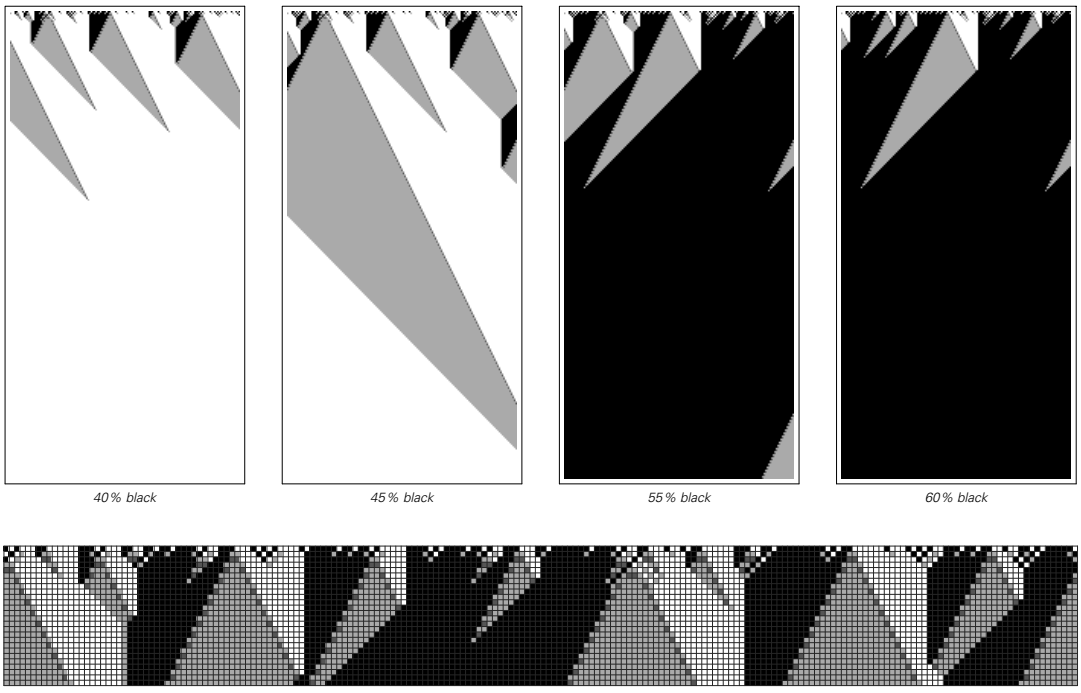


A one-dimensional cellular automaton that shows a discrete change in behavior when the properties of its initial conditions are continuously changed. If the initial density of black cells is less than 50%, then only white stripes ultimately survive. But as soon as the density increases above 50%, the white stripes disappear, and black stripes dominate. The underlying rule for the cellular automaton shown takes the new color of a cell to be the color of its right neighbor if the cell is black and its left neighbor if the cell is white. (This corresponds to rule 184 in the scheme described on page 53.)

When the initial density of black cells has any value less than 50%, only white stripes ever survive. But as soon as the initial density increases above 50%, a discrete transition occurs, and it is black stripes, rather than white, that survive.

The pictures on the facing page show another example of the same basic phenomenon. When the initial density of black cells is less than 50%, all regions of black eventually disappear, and the system becomes completely white. But as soon as the density increases above 50%, the behavior suddenly changes, and the system eventually becomes completely black.

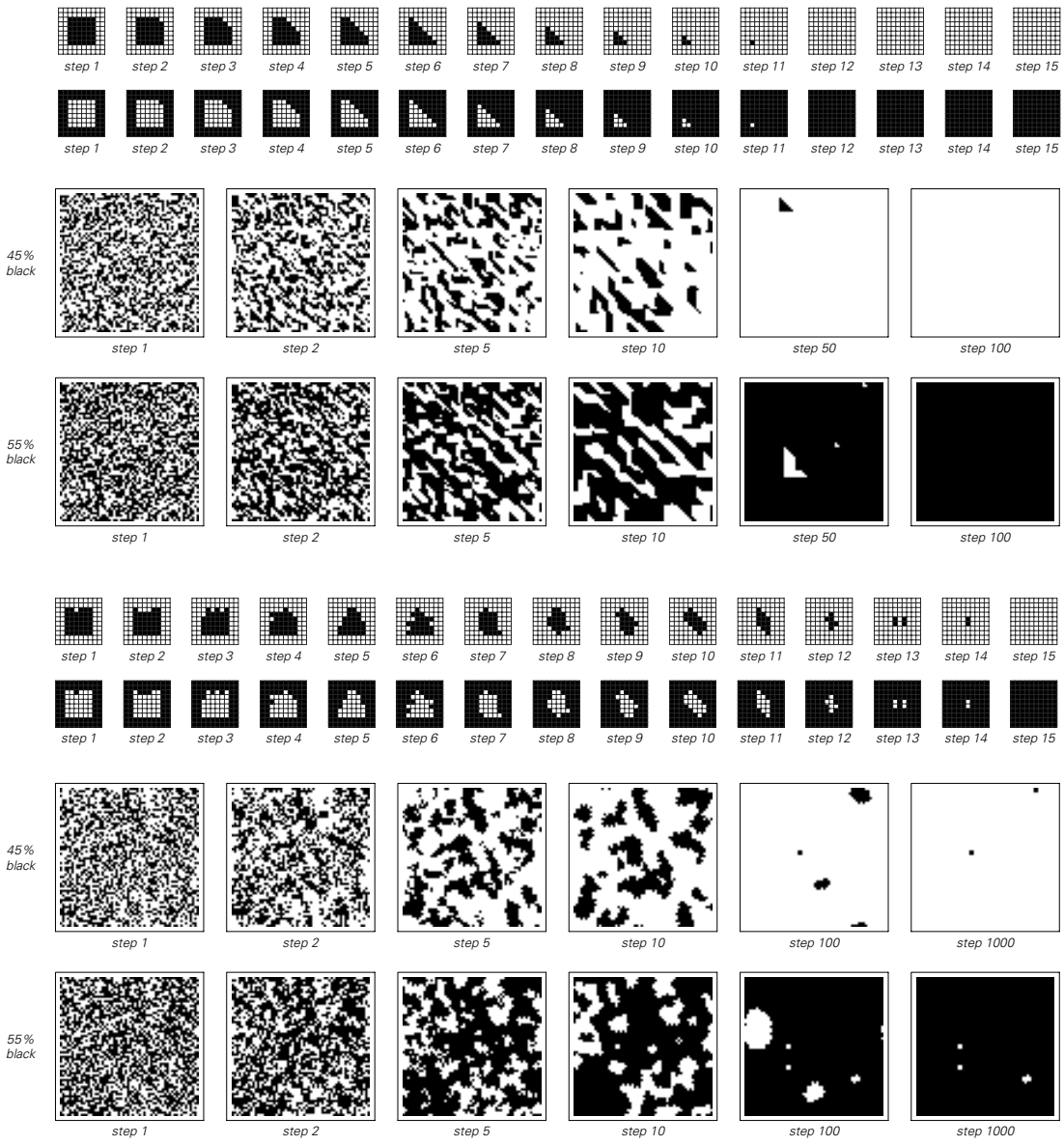
It turns out that such discrete transitions are fairly rare among one-dimensional cellular automata, but in two and more dimensions



A one-dimensional cellular automaton in which the density of black cells obtained after a large number of steps changes discretely when the initial density of black cells is continuously increased. With an initial density below 50%, regions of black always eventually disappear. But as soon as the density is increased above 50%, regions of black progressively expand, eventually taking over the whole system. The underlying rule allows four possible colors for each cell. The rule is set up so that whenever a region of black occurs to the left of a region of white, an expanding region of gray appears in between. The crucial point is then that if the region of white is narrower than the region of black, then the gray will reach the edge of the white before it reaches the edge of the black. And when this happens, the black expands and the gray gradually tapers away.

they become increasingly common. The pictures on the next page show two examples—the second corresponding to a rule that we saw in a different context at the end of the previous section.

In both examples, what essentially happens is that in regions where there is an excess of black over white, an increasingly large fraction of cells become black, while in regions where there is an excess of white over black, the reverse happens. And so long as the boundaries of the regions do not get stuck—as happens in many one-dimensional cellular automata—the result is that whichever color was initially more common eventually takes over the whole system.

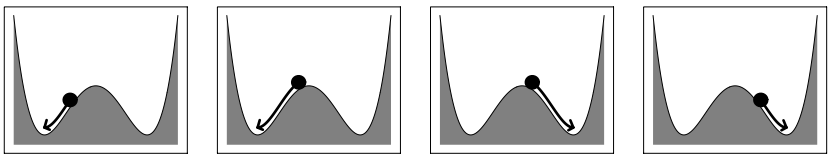


Two examples of two-dimensional cellular automata that show discrete transitions in behavior when the density of initial black cells is continuously varied. In the top rule, the new color of a particular cell is found simply by looking at that cell and its immediate neighbors above and to the right. If two or more of these three cells are black, then the new color is black; otherwise it is white. The pictures in the middle above show that with this rule blocks of opposite color are progressively destroyed, so that whichever color was initially more common eventually dominates completely. The bottom rule above is exactly the same as was shown on page 336. Whichever color was initially more common again eventually dominates, though with this rule it takes somewhat longer for this to occur.

In most cellular automata, the behavior obtained after a long time is either largely independent of the initial density, or varies quite smoothly with it. But the special feature of the cellular automata shown on the facing page is that they have two very different stable states—either all white or all black—and when one changes the initial density a discrete transition occurs between these two states.

One might think that the existence of such a discrete transition must somehow be associated with the discrete nature of the underlying cellular automaton rules. But it turns out that it is also possible to get such transitions in systems that have continuous underlying rules.

The pictures below show a standard very simple example of how this can happen. If one starts to the left of the center hump, then the ball will always roll into the left-hand minimum. But if one progressively changes the initial position of the ball, then when one passes the center a discrete transition occurs, and the ball instead rolls into the right-hand minimum.



A standard simple example of a continuous system in which there is a discrete change in behavior as a consequence of a continuous change in initial conditions. When the ball starts anywhere to the left of the center line, it rolls into the left-hand minimum. But if instead it starts on the right, then it rolls into the right-hand minimum. There are many systems in nature that follow the same general form of mathematical equations as those that describe the energy and motion of the ball.

Thus even though the mathematical equations which govern the motion of the ball have a simple continuous form, the behavior they produce still involves a discrete transition. And while this particular example may seem contrived, it turns out that essentially the same mathematical equations also occur in many other situations—such as the evolution of chemical concentrations in various chemical reactions.

And whenever such equations arise, they inevitably lead to a limited number of stable states for the system, with discrete transitions occurring between these states when the parameters of the system are varied.

So even if a system at some level follows continuous rules it is still possible for the system to exhibit discrete overall behavior. And in fact it is quite common for such behavior to be one of the most obvious features of a system—which is why discrete systems like cellular automata end up often being the most appropriate models.

The Problem of Satisfying Constraints

One feature of programs is that they immediately provide explicit rules that can be followed to determine how a system will behave. But in traditional science it is common to try to work instead with constraints that are merely supposed implicitly to force certain behavior to occur.

At the end of Chapter 5 I gave some examples of constraints, and I showed that constraints do exist that can force quite complex behavior to occur. But despite this, my strong suspicion is that of all the examples of complex behavior that we see in nature almost none can in the end best be explained in terms of constraints.

The basic reason for this is that to work out what pattern of behavior will satisfy a given constraint usually seems far too difficult for it to be something that happens routinely in nature.

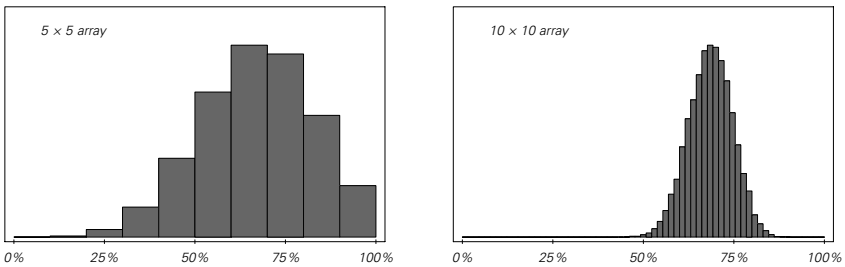
Many types of constraints—including those in Chapter 5—have the property that given a specific pattern it is fairly easy to check whether the pattern satisfies the constraints. But the crucial point is that this fact by no means implies that it is necessarily easy to go from the constraints to find a pattern that satisfies them.

The situation is quite different from what happens with explicit evolution rules. For if one knows such rules then these rules immediately yield a procedure for working out what behavior will occur. Yet if one only knows constraints then such constraints do not on their own immediately yield any specific procedure for working out what behavior will occur.

In principle one could imagine looking at every possible pattern, and then picking out the ones that satisfy the constraints. But even with a 10×10 array of black and white squares, the number of possible patterns is already 1,267,650,600,228,229,401,496,703,205,376. And with a

20 × 20 array this number is larger than the total number of particles in the universe. So it seems quite inconceivable that systems in nature could ever carry out such an exhaustive search.

One might imagine, however, that if such systems were just to try patterns at random, then even though incredibly few of these patterns would satisfy any given constraint exactly, a reasonable number might at least still come close. But typically it turns out that even this is not the case. And as an example, the pictures below show what fraction of patterns chosen at random have a given percentage of squares that violate the constraints described on page 211.



The fraction of all possible patterns in which a certain percentage of squares violate the constraints discussed on page 211. Only a handful of patterns satisfy the constraints exactly (so that 0% of the squares are wrong). For large arrays, the vast majority of possible patterns have about 70% of the squares wrong.

For the majority of patterns around 70% of the squares turn out to violate the constraints. And in a 10 × 10 array the chance of finding a pattern where the fraction of squares that violate the constraints is even less than 50% is only one in a thousand, while the chance of finding a pattern where the fraction is less than 25% is one in four trillion.

And what this means is that a process based on picking patterns at random will be incredibly unlikely to yield results that are even close to satisfying the constraints.

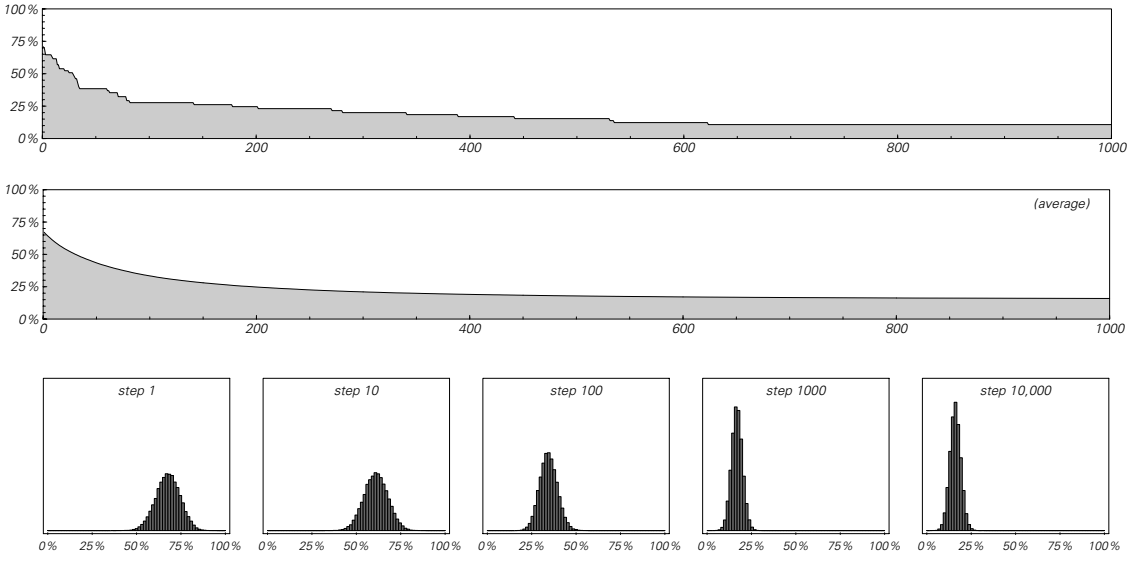
So how can one do better? A common approach used both in natural systems and in practical computing is to have some form of iterative procedure, in which one starts from a pattern chosen at

random, then progressively modifies the pattern so as to make it closer to satisfying the constraints.

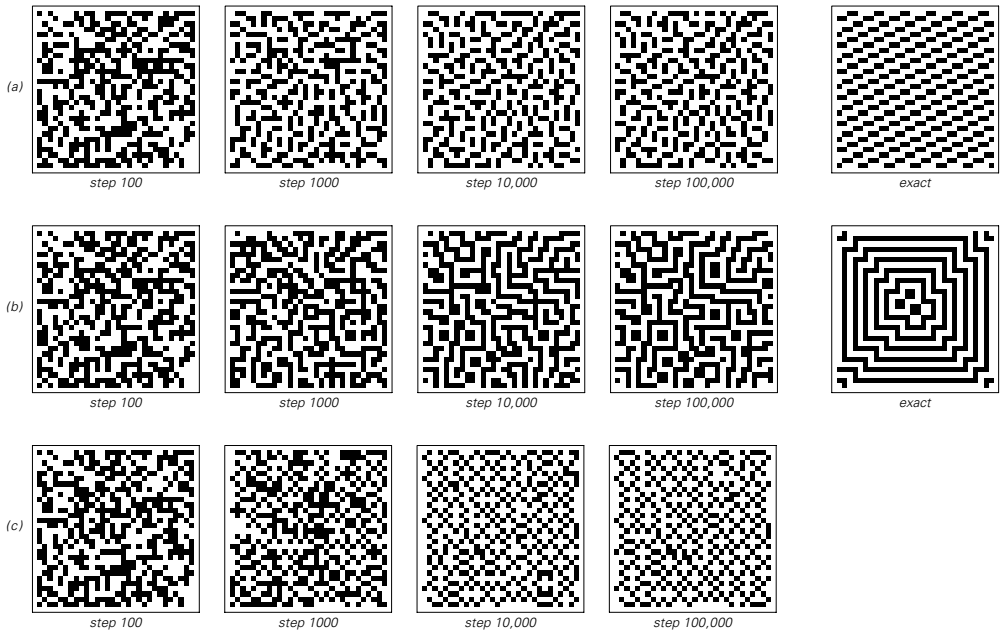
As a specific example consider taking a series of steps, and at each step picking a square in the array discussed above at random, then reversing the color of this square whenever doing so will not increase the total number of squares in the array that violate the constraints.

The picture below shows results obtained with this procedure. For the first few steps, there is rapid improvement. But as one goes on, one sees that the rate of improvement gets slower and slower. And even after a million steps, it turns out that 15% of the squares in a 10×10 array will on average still not satisfy the constraints.

In practical situations this kind of approximate result can sometimes be useful, but the pictures at the top of the facing page show that the actual patterns obtained do not look much at all like the exact results that we saw for this system in Chapter 5.



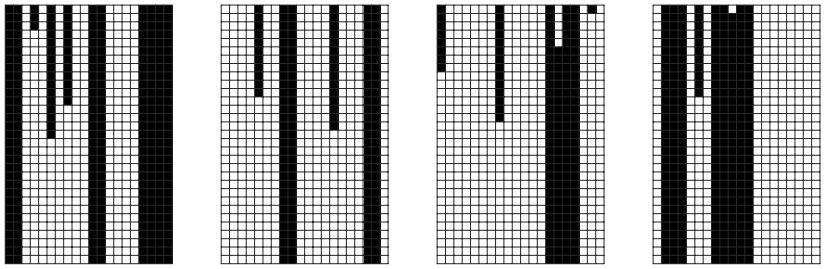
The results of a procedure intended to produce patterns that get progressively closer to satisfying the constraints described on page 211. The procedure starts with a randomly chosen pattern, then at each step picks a square in the pattern at random, and reverses the color of this square whenever doing so does not increase the total number of squares in the pattern that violate the constraints. The top picture shows one particular run of this procedure. The second picture shows the average behavior obtained from many runs. And finally, the bottom picture shows how the fraction of patterns with different percentages of squares violating the constraints changes as the procedure progresses. In all cases 10×10 patterns are used.



Patterns generated by using the same procedure as in the previous picture but with three different sets of constraints. Case (a) uses the same constraints as in the previous picture, (b) requires every black square and every white square to have exactly two adjacent black squares, and (c) requires every black square to have 3 adjacent black squares and 1 white square, and every white square to have 4 adjacent white squares. In cases (a) and (b) it is possible to satisfy the constraints exactly; in case (c) it is not. The pictures show the evolution of a 30 × 30 array, which is nearly 10 times the area of the array shown in the previous picture. Although the fraction of squares that violate the constraints is less than 20% after 100,000 steps, the overall patterns still do not look much like the exact results.

So why does the procedure not work better? The problem turns out to be a rather general one. And as a simple example, consider a line of black and white squares, together with the constraint that each square should have the same color as its right-hand neighbor. This constraint will be satisfied only if every square has the same color—either black or white. But to what extent will an iterative procedure succeed in finding this solution?

As a first example, consider a procedure that at each step picks a square at random, then reverses its color whenever doing so reduces the total number of squares that violate the constraint. The pictures at the top of the next page show what happens in this case. The results are

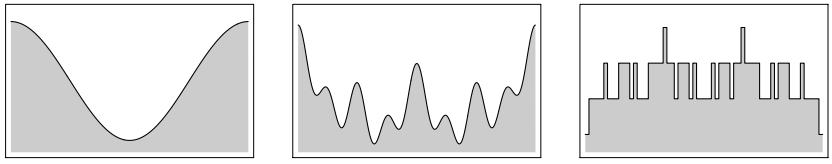


Results of four tries at applying an iterative procedure to find configurations which satisfy the simple constraint that every square should be the same color as the square to its right. (The squares are assumed to be arranged cyclically, so that the right neighbor of the rightmost square is the leftmost square.) The procedure starts from a random configuration of squares, and then at each step picks a square at random, then reverses the color of this square whenever doing so reduces the total number of squares that violate the constraint. The only configurations that ultimately satisfy the constraints are all white and all black. But the procedure gets stuck long before it reaches these configurations. The problem is that for any block more than one square across changing the color of a square at either end will not reduce the total number of squares that violate the constraint. And as a result, such blocks remain fixed and cannot disappear.

remarkably poor: instead of steadily evolving to all black or all white, the system quickly gets stuck in a state that contains regions of different colors.

And as it turns out, this kind of behavior is not uncommon among iterative procedures; indeed it is even seen in such simple cases as trying to find the lowest point on a curve. The most obvious iterative procedure to use for such a problem involves taking a series of small steps, with the direction of each step being chosen so as locally to go downhill.

And indeed for the first curve shown below, this procedure works just fine, and quickly leads to the lowest point. But for the second



Three examples of curves. In the first case, the most obvious mechanical or mathematical procedure of continually going downhill will successfully lead one to the lowest point. But in the other two cases, this procedure will usually end up getting stuck at a local minimum. This is the basic phenomenon which makes it difficult to find patterns that satisfy constraints exactly using a procedure that is based on progressive improvement. The third picture above is a representation of the kind of curve that arises in almost all discrete systems based on constraints.

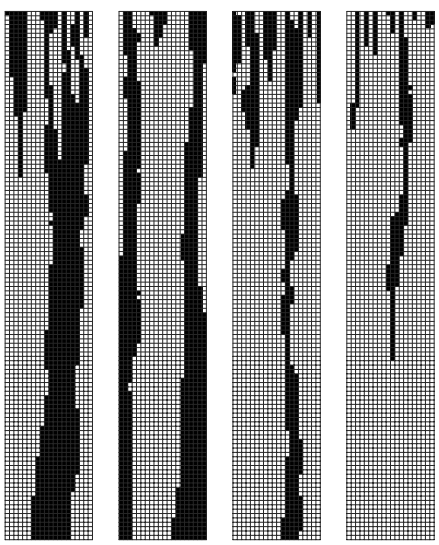
curve, the procedure will already typically not work; it will usually get stuck in one of the local minima and never reach a global minimum.

And for discrete systems involving, say, just black and white squares, it turns out to be almost inevitable that the curves which arise have the kind of jagged form shown in the third picture at the bottom of the facing page. So this has the consequence that a simple iterative procedure that always tries to go downhill will almost invariably get stuck.

How can one avoid this? One general strategy is to add randomness, so that in essence one continually shakes the system to prevent it from getting stuck. But the details of how one does this tend to have a great effect on the results one gets.

The procedure at the top of the facing page already in a sense involved randomness, for it picked a square at random at each step. But as we saw, with this particular procedure the system can still get stuck.

Modifying the procedure slightly, however, can avoid this. And as an example the pictures below show what happens if at each step one reverses the color of a random square not only if this will decrease the total number of squares violating the constraints, but also if it leaves this number the same. In this case the system never gets permanently stuck, and instead will always eventually evolve to satisfy the constraints.



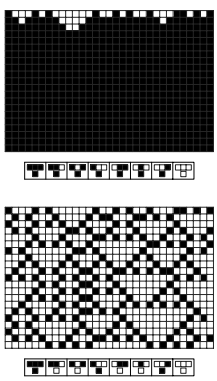
Results from a slight modification to the procedure used in the picture at the top of the facing page. A random square is again picked at each step. But now the color of that square is reversed not only if doing so actually changes the total number of squares that violate the constraint, but also if it leaves this number the same. With this procedure, evolution from any initial condition can visit every possible configuration, so that the configurations which satisfy the constraints will at least eventually be reached.

But this process may still take a very long time. And indeed in the two-dimensional case discussed earlier in this section, the number of steps required can be quite astronomically long.

So can one speed this up? The more one knows about a particular system, the more one can invent tricks that work for that system. But usually these turn out to lead only to modest speedups, and despite various hopes over the years there seem in the end to be no techniques that work well across any very broad range of systems.

So what this suggests is that even if in some idealized sense a system in nature might be expected to satisfy certain constraints, it is likely that in practice the system will actually not have a way to come even close to doing this.

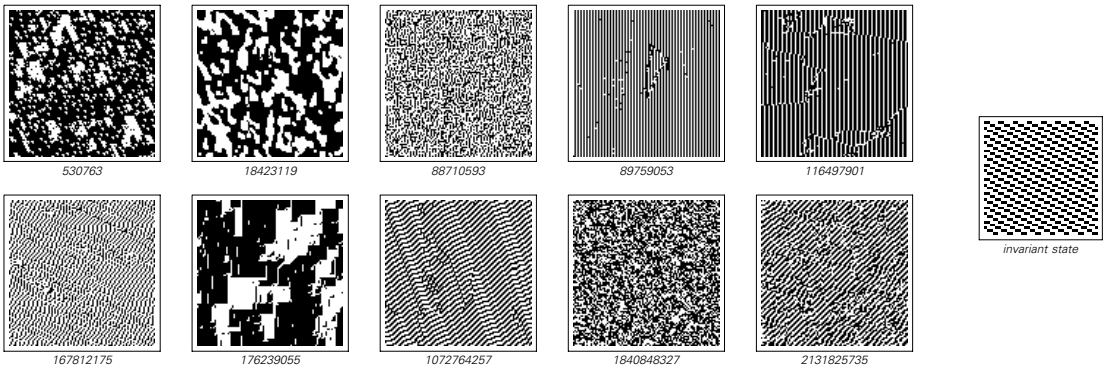
In traditional science the notion of constraints is often introduced in an attempt to summarize the effects of evolution rules. Typically the idea is that after a sufficiently long time a system should be found only in states that are invariant under the application of its evolution rules. And quite often it turns out that one can show that any states that are invariant in this way must satisfy fairly simple constraints. But the problem is that except in cases where the behavior as a whole is very simple it tends not to be true that systems in fact evolve to strictly invariant states.



Two of the 28 elementary cellular automata whose only invariant states are uniform in color. In the first case one of these invariant states is always reached; in the second it is not.

The two cellular automata on the left both have all white and all black as invariant states. And in the first case, starting from random initial conditions, the system quickly settles down to the all black invariant state. But in the second case, nothing like this happens, and instead the system continues to exhibit complicated and seemingly random behavior forever.

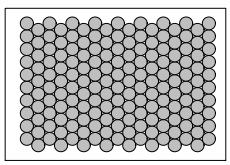
The two-dimensional patterns that arise from the constraints at the end of Chapter 5 all turn out to correspond to invariant states of various two-dimensional cellular automata. And so for example the pattern of page 211 is found to be the unique invariant state for 572,522 of the 4,294,967,296 possible five-neighbor cellular automaton rules. But if one starts these rules from random initial conditions, one typically never gets the pattern of page 211. Instead, as the pictures at the top of the facing page show, one sees a variety of patterns that very



Typical behavior of two-dimensional cellular automata that leave only the pattern on the right invariant. The results shown come from 500 steps of evolution starting from random initial conditions. In no case does the global behavior seen come even close to satisfying the simple constraints that determine the invariant state.

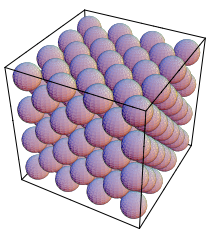
much more reflect explicit rules of evolution than the constraint associated with the invariant state.

So what about actual systems in physics? Do they behave any differently? As one example, consider a large number of circular coins pushed together on a table. One can think of such a system as having an invariant state that satisfies the constraint that the coins should be packed as densely as possible. For identical coins this constraint is satisfied by the simple repetitive pattern shown on the right. And it turns out that in this particular case this pattern is quickly produced if one actually pushes coins together on a table.



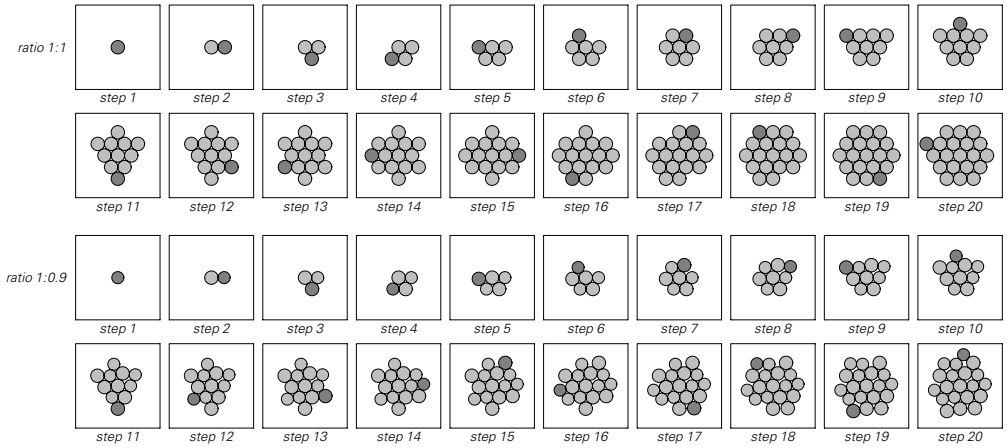
The densest packing of identical circles in the plane. Each circle is surrounded by six others.

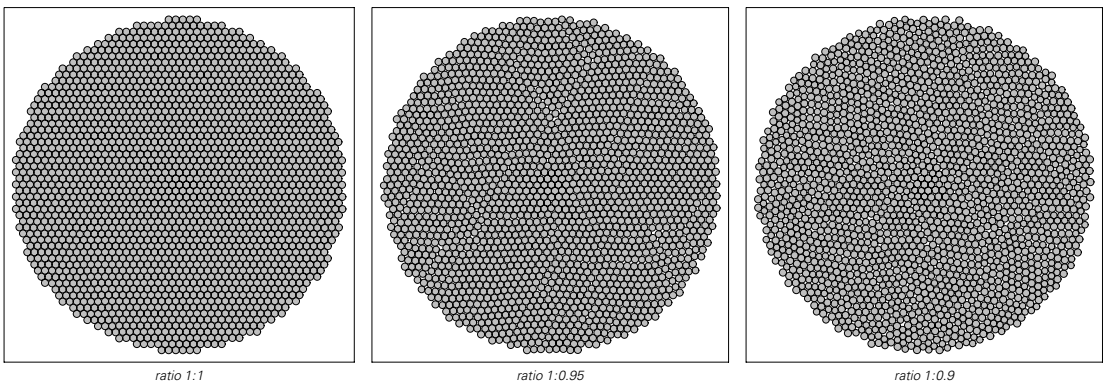
But with balls in three dimensions the situation is quite different. In this case the constraint of densest packing is known to be satisfied when the balls are laid out in the simple repetitive way shown on the right. But if one just tries pushing balls together they almost always get stuck, and never take on anything like the arrangement shown. And if one jiggles the balls around one still essentially never gets this arrangement. Indeed, the only way to do it seems to be to lay the balls down carefully one after another.



The densest packing of identical spheres in three-dimensional space. Each sphere is surrounded by 12 others.

In two dimensions similar issues arise as soon as one has coins of more than one size. Indeed, even with just two sizes, working out how to satisfy the constraint of densest packing is already so difficult that in most cases it is still not known what configuration does it.





Patterns obtained by successively laying down circles in such a way that the center of each new circle is as close as possible to the center of the first circle. Except in the very first case, the extent to which these represent the densest possible packings is not clear, and indeed it is quite possible that in most such actual packings circles of different sizes are just separated into several uniform regions.

The pictures on the facing page show what happens if one starts with a single circle, then successively adds new circles in such a way that the center of each one is as close to the center of the first circle as possible. When all circles are the same size, this procedure yields a simple repetitive pattern. But as soon as the circles have significantly different sizes, the pictures on the facing page show that this procedure tends to produce much more complicated patterns—which in the end may or may not have much to do with the constraint of densest packing.

One can look at all sorts of other physical systems, but so far as I can tell the story is always more or less the same: whenever there is behavior of significant complexity its most plausible explanation tends to be some explicit process of evolution, not the implicit satisfaction of constraints.

One might still suppose, however, that the situation could be different in biological systems, and that somehow the process of natural selection might produce forms that are successfully determined by the satisfaction of constraints.

But what I strongly believe, as I discuss in the next chapter, is that in the end, much as in physical systems, only rather simple forms can actually be obtained in this way, and that when more complex forms are seen they once again tend to be associated not with constraints but rather with the effects of explicit evolution rules—mostly those governing the growth of an individual organism.

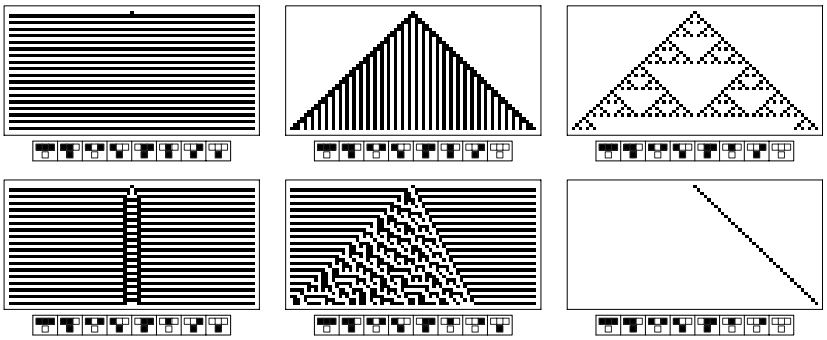
Origins of Simple Behavior

There are many systems in nature that show highly complex behavior. But there are also many systems that show rather simple behavior—most often either complete uniformity, or repetition, or nesting.

And what we have found in this book is that programs are very much the same: some show highly complex behavior, while others show only rather simple behavior.

Traditional intuition might have made one assume that there must be a direct correspondence between the complexity of observed behavior and the complexity of underlying rules. But one of the central discoveries of this book is that in fact there is not.

For even programs with some of the very simplest possible rules yield highly complex behavior, while programs with fairly complicated rules often yield only rather simple behavior. And indeed, as we have seen many times in this book, and as the pictures below illustrate, even rules that are extremely similar can produce quite different behavior.



A sequence of elementary cellular automata whose rules differ from one to the next only at one position (a Gray code sequence). Despite the similarity of their rules, the overall behavior of these cellular automata differs considerably.

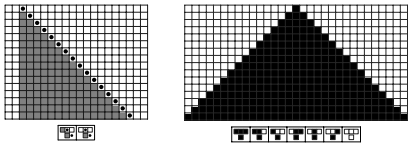
If one just looks at a rule in its raw form, it is usually almost impossible to tell much about the overall behavior it will produce. But in cases where this behavior ends up being simple, one can often recognize in it specific mechanisms that seem to be at work.

If the behavior of a system is simple, then this inevitably means that it will have many regularities. And usually there is no definite way to say which of these regularities should be considered causes of what one sees, and which should be considered effects.

But it is still often useful to identify simple mechanisms that can at least serve as descriptions of the behavior of a system.

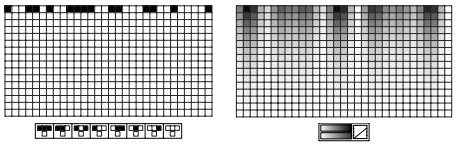
In many respects the very simplest possible type of behavior in any system is pure uniformity. And uniformity in time is particularly straightforward, for it corresponds just to no change occurring in the evolution of a system. But uniformity in space is already slightly more complicated, and indeed there are several different mechanisms that can be involved in it. A rather straightforward one, illustrated in the pictures

below, is that some process can start at one point in space and then progressively spread, doing the same thing at every point it reaches.



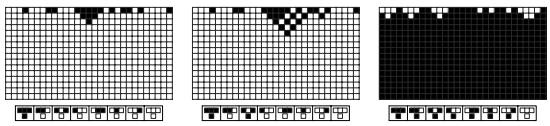
Homogenous growth from a single point is one straightforward way that uniformity in space can be produced, here illustrated in a mobile automaton and a cellular automaton.

Another mechanism is that every part of a system can evolve completely independently to the same state, as in the pictures below.



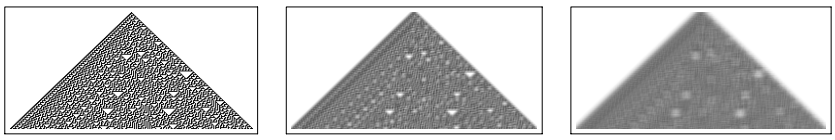
Uniformity in space can be achieved almost trivially if each element in a system independently evolves to the same state.

A slightly less straightforward mechanism is illustrated in the pictures below. Here different elements in the system do interact, but the result is still that all of them evolve to the same state.



Class 1 cellular automata that exhibit evolution to a uniform state, as discussed in Chapter 6.

So far all the mechanisms for uniformity I have mentioned involve behavior that is in a sense simple at every level. But in nature uniformity often seems to be associated with quite complex microscopic behavior. Most often what happens is that on a small scale a system exhibits randomness, but on a larger scale this randomness averages out to leave apparent uniformity, as in the pictures below.

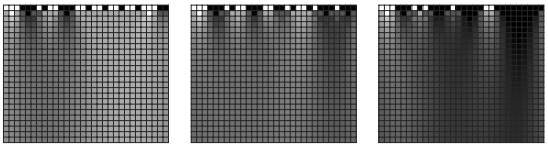


Averaging out small-scale randomness yields apparent uniformity, as shown here for a rule 30 pattern.

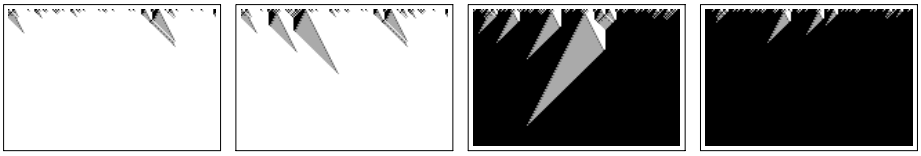
It is common for uniform behavior to be quite independent of initial conditions or other input to a system. But sometimes different uniform behavior can be obtained with different input.

One way this can happen, illustrated in the pictures below, is for the system to conserve some quantity—such as total density of black—and for this quantity to end up being spread uniformly throughout the system by its evolution.

With each cell at each step having a gray level that is the average of its predecessor and its two neighbors the total amount of black is conserved, but eventually becomes spread uniformly throughout the system.



An alternative is that the system may always evolve to certain specific uniform phases, but the choice of which phase may depend on the total value of some quantity, as in the pictures below.

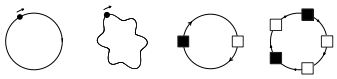


With different initial conditions this cellular automaton from page 339 can evolve either to uniform white or uniform black. Such discrete transitions are somewhat less common in one dimension than elsewhere.

Constraints are yet another basis for uniformity. And as a trivial example, the constraint in a line of black or white cells that every cell should be the same color as both its neighbors immediately implies that the whole line must be either uniformly black or uniformly white.

Beyond uniformity, repetition can be considered the next-simplest form of behavior. Repetition in time corresponds just to a system repeatedly returning to a particular state.

This can happen if, for example, the behavior of a system in effect follows some closed curve such as a circle which always leads back to the same point. And in general, in any system with definite rules that only ever visits a limited number of states, it is

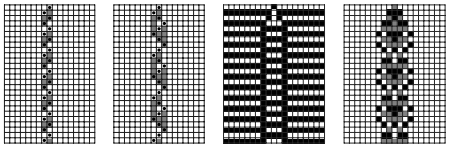


The behavior of a system will be repetitive in time whenever it effectively follows a closed curve—either literally in space, or in terms of states that it visits.

inevitable—as discussed on page 255 and illustrated above—that the behavior of the system will eventually repeat.

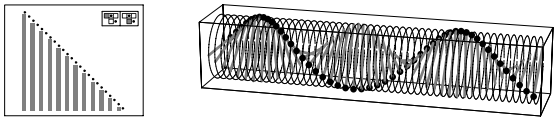
In some cases the basic structure of a system may allow only a limited number of possible states. But in other cases what happens is instead just that the actual evolution of a system never reaches more than a limited number of states.

Often it is very difficult to predict whether this will be so just by looking at the underlying rules. But in a system like a cellular automaton the typical reason for it is just that in the end effects never spread beyond a limited region, as in the examples shown below.



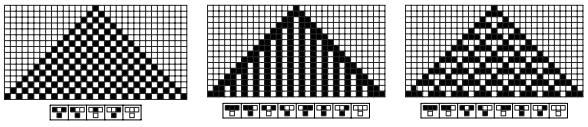
Examples of behavior in mobile automata and cellular automata that remains localized to a limited region and thus always eventually repeats.

Given repetition in time, repetition in space will follow whenever elements that repeat systematically move in space. The pictures below show two cases of this, with the second picture illustrating the notion of waves that is common in traditional physics.



Examples where repetition in time leads directly to repetition in space. The second picture shows standard wave motion.

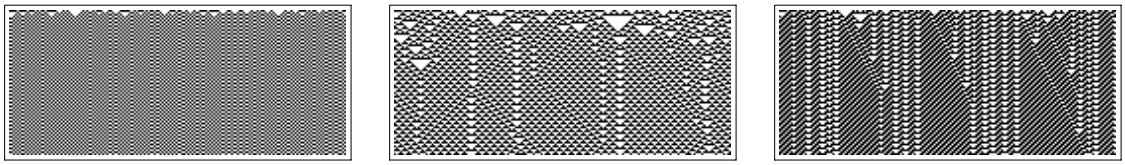
Growth from a simple seed can also readily lead to repetition in both space and time, as in the pictures below.



Cellular automata in which a repetitive pattern in both space and time is generated by evolution from a simple seed.

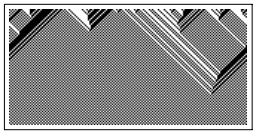
But what about random initial conditions? Repetition in time is still easy to achieve—say just by different parts of a system behaving independently. But repetition in space is slightly more difficult to achieve. For even if localized domains of repetition form, they need to have some mechanism for combining together.

And the walls between different domains often end up not being mobile enough to allow this to happen, as in the examples below.



rule 50 rule 54 rule 62

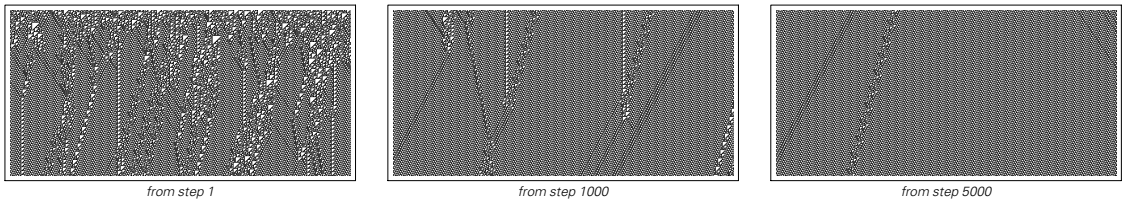
Cellular automata in which domains of repetitive behavior form, but in which walls typically remain forever between these domains.



A cellular automaton (rule 184) in which domains quickly combine to make the whole system repetitive in space.

But there are certainly cases—in one dimension and particularly above—where different domains do combine, and exact repetition is achieved. Sometimes this happens quickly, as in the picture on the left.

But in other cases it happens only rather slowly. An example is rule 110, in which repetitive domains form with period 14 in space and 7 in time, but as the picture below illustrates, the localized structures which separate these domains take a very long time to disappear.



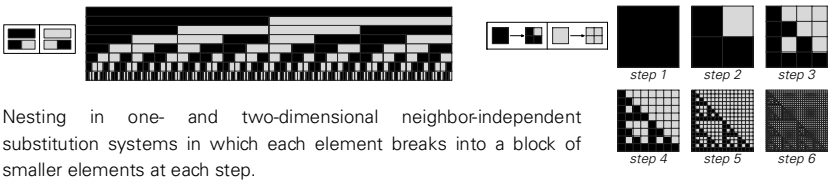
from step 1 from step 1000 from step 5000

The behavior of rule 110 starting from random initial conditions. Domains of repetitive behavior are formed, which in most cases gradually combine as the localized structures which separate them disappear.

As we saw at the end of Chapter 5, many systems based on constraints also in principle yield repetition—though from the discussion of the previous section it seems likely that this is rarely a good explanation for actual repetition that we see in nature.

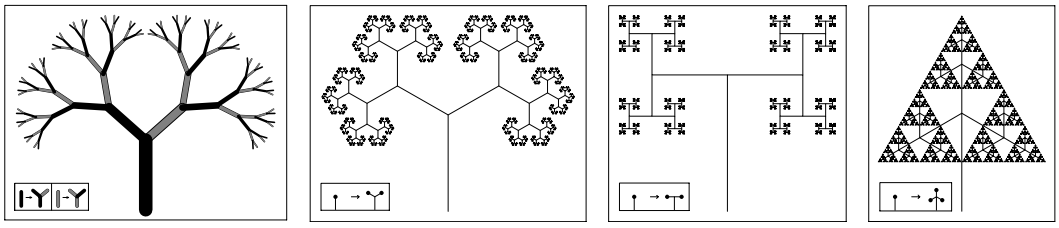
Beyond uniformity and repetition, the one further type of simple behavior that we have often encountered in this book is nesting. And as with uniformity and repetition, there are several quite different ways that nesting seems to arise.

Nesting can be defined by thinking in terms of splitting into smaller and smaller elements according to some fixed rule. And as the pictures below illustrate, nested patterns are generated very directly in substitution systems by each element successively splitting explicitly into blocks of smaller and smaller elements.



Nesting in one- and two-dimensional neighbor-independent substitution systems in which each element breaks into a block of smaller elements at each step.

An essentially equivalent process involves every element branching into smaller and smaller elements and eventually forming a tree-like structure, as in the pictures below.



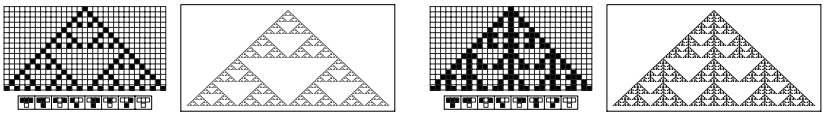
Nested patterns generated by simple branching processes. (Compare page 406.)

So what makes a system in nature operate in this way? Part of it is that the same basic rules must apply regardless of physical scale. But on its own this would be quite consistent with various kinds of uniform or spiral growth, and does not imply that there will be what we usually think of as nesting. And indeed to get nesting seems to require that there also be some type of discrete splitting or branching process in which several distinct elements arise from an individual element.

A somewhat related source of nesting relevant in many mathematical systems is the nested pattern formed by the digit sequences of successive numbers, as illustrated on page 117.

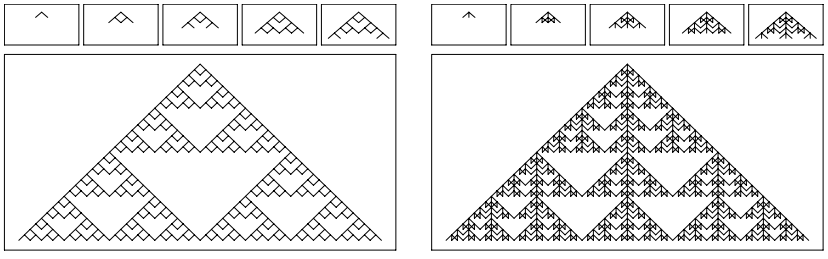
But in general nesting need not just arise from larger elements being broken down into smaller ones: for as we have discovered in this book it can also arise when larger elements are built up from smaller ones—and indeed I suspect that this is its more common origin in nature.

As an example, the pictures below show how nested patterns with larger and larger features can be built up by starting with a single black cell, and then following simple additive cellular automaton rules.



Nested patterns built by the evolution of the rule 90 and rule 150 additive cellular automata starting from a single black cell.

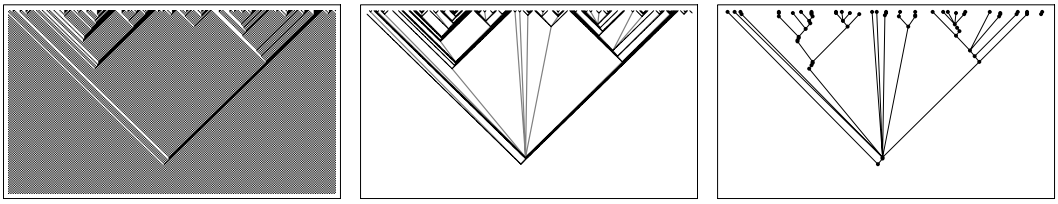
It turns out that the very same patterns can also be produced—as the pictures below illustrate—by processes in which new branches form at regular intervals, and annihilate when any pair of them collide.



Nested patterns obtained by processes in which either two or three branches are formed at regular intervals, and annihilate when any pair of them collide.

But what about random initial conditions? Can nesting also arise from these? It turns out that it can. And the basic mechanism is typically some kind of progressive annihilation of elements that are initially distributed randomly.

The pictures below show an example, based on the rule 184 cellular automaton. Starting from random initial conditions this rule yields a collection of stripes which annihilate whenever they meet, leading to a sequence of progressively larger nested regions.

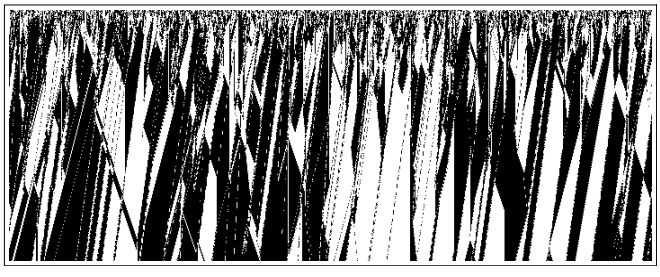


The generation of a nested pattern by rule 184 starting from random initial conditions. The pattern consists of a collection of stripes, highlighted in the second picture, which form the tree-like structure shown in the third picture. The initial condition used has exactly equal numbers of black and white cells, causing all the stripes eventually to annihilate.

And as the pictures show, these regions form a pattern that corresponds to a random tree that builds up from its smallest branches, much in the way that a river builds up from its tributaries.

Nesting in rule 184 is easiest to see when the initial conditions contain exactly equal numbers of black and white cells, so that the numbers of left and right stripes exactly balance, and all stripes eventually annihilate. But even when the initial conditions are such that some stripes survive, nested regions are still formed by the stripes that do annihilate. And indeed in essentially any system where there are domains that grow fairly independently and then progressively merge the same basic overall nesting will be seen.

As an example, the picture below shows the rule 110 cellular automaton evolving from random initial conditions. The picture

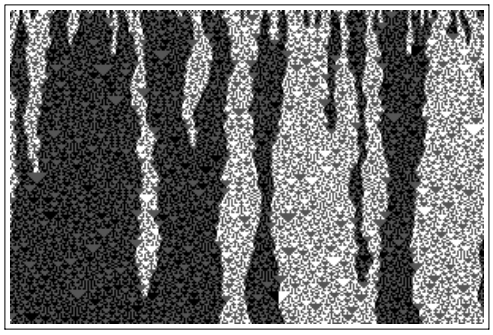


A highly compressed representation of the evolution of rule 110 from random initial conditions in which only the first cell in every 14 × 7 block is sampled.

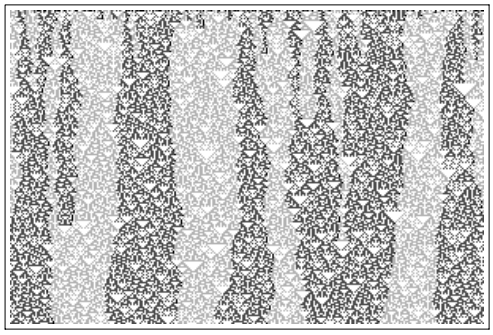
samples just the first cell in every 14 × 7 block of cells, making each domain of repetitive behavior stand out as having a uniform color.

In the detailed behavior of the various localized structures that separate these domains of repetitive behavior there is all sorts of complexity. But what the picture suggests is that at some rough overall level these structures progressively tend to annihilate each other, and in doing so form an approximate nested pattern.

It turns out that this basic process is not restricted to systems which produce simple uniform or repetitive domains. And the pictures below show for example cases where the behavior inside each domain is quite random.



k=3 totalistic code 1893



elementary rule 18 (compressed)

Examples involving domains containing apparent randomness. In the second picture, each element shown represents a 2 × 2 block of original cells. In both cases, the boundaries between domains appear to follow random walks, annihilating when they meet and thus forming a nested overall pattern.

Instead of following simple straight lines, the boundaries of these domains now execute seemingly random walks. But the fact that they annihilate whenever they meet once again tends to lead to an overall nested pattern of behavior.

So what about systems based on constraints? Can these also lead to nesting? In Chapter 5 I showed that they can. But what I found is that whereas at least in principle both uniformity and repetition can be forced fairly easily by constraints, nesting usually cannot be.

At the outset, one might have thought that there would be just one definite mechanism for each type of simple behavior. But what we

have seen in this section is that in fact there are usually several apparently quite different mechanisms possible.

Often one can identify features in common between the various mechanisms for any particular kind of behavior. But typically these end up just being inevitable consequences of the fact that some specific kind of behavior is being produced.

And so, for example, one might notice that most mechanisms for nesting can at some level be viewed as involving hierarchies in which higher components affect lower ones, but not the other way around. But in a sense this observation is nothing more than a restatement of a property of nesting itself.

So in the end one can indeed view most of the mechanisms that I have discussed in this section as being in some sense genuinely different. Yet as we have seen all of them can be captured by quite simple programs. And in Chapter 12 I will discuss how this is related to the fact that so few fundamentally different types of overall behavior ultimately seem to occur.

Mechanisms in Programs and Nature

Universality of Behavior

■ **History.** That very different natural and artificial systems can show similar forms has been noted for many centuries. Informal studies have been done by a whole sequence of architects interested both in codifying possible forms and in finding ways to make structures fit in with nature and with our perception of it. Beginning in the Renaissance the point has also been noted by representational and decorative artists, most often in the context of developing a theory of the types of forms to be studied by students of art. The growth of comparative anatomy in the 1800s led to attempts at more scientific treatments, with analogies between biological and physical systems being emphasized particularly by D'Arcy Thompson in 1917. Yet despite all this, the phenomenon of similarity between forms remained largely a curiosity, discussed mainly in illustrated books with no clear basis in either art or science. In a few cases (such as work by Peter Stevens in 1974) general themes were however suggested. These included for example symmetry, the golden ratio, spirals, vortices, minimal surfaces, branching patterns, and—since the 1980s—fractals. The suggestion is also sometimes made that we perceive a kind of harmony in nature because we see only a limited number of types of forms in it. And particularly in classical architecture the idea is almost universally used that structures will seem more comfortable to us if they repeat in ornament or otherwise forms with which we have become familiar from nature. Whenever a scientific model has the same character for different systems this means that the systems will tend to show similar forms. And as models like cellular automata capable of dealing with complexity have become more widespread it has been increasingly popular to show that they can capture similar forms seen in very different systems.

Three Mechanisms for Randomness

■ **Page 299 · Definition.** How randomness can be defined is discussed at length on page 552.

■ **History.** In antiquity, it was often assumed that all events must be governed by deterministic fate—with any apparent randomness being the result of arbitrariness on the part of the gods. Around 330 BC Aristotle mentioned that instead randomness might just be associated with coincidences outside whatever system one is looking at, while around 300 BC Epicurus suggested that there might be randomness continually injected into the motion of all atoms. The rise of emphasis on human free will (see page 1135) eroded belief in determinism, but did not especially address issues of randomness. By the 1700s the success of Newtonian physics seemed again to establish a form of determinism, and led to the assumption that whatever randomness was actually seen must reflect lack of knowledge on the part of the observer—or particularly in astronomy some form of error of measurement. The presence of apparent randomness in digit sequences of square roots, logarithms, numbers like π, and other mathematical constructs was presumably noticed by the 1600s (see page 911), and by the late 1800s it was being taken for granted. But the significance of this for randomness in nature was never recognized. In the late 1800s and early 1900s attempts to justify both statistical mechanics and probability theory led to ideas that perfect microscopic randomness might somehow be a fundamental feature of the physical world. And particularly with the rise of quantum mechanics it came to be thought that meaningful calculations could be done only on probabilities, not on individual random sequences. Indeed, in almost every area where quantitative methods were used, if randomness was observed, then either a different system was studied, or efforts were made to remove the randomness by averaging or some other statistical method. One case where there was occasional discussion of origins of randomness from at least

the early 1900s was fluid turbulence (see page 997). Early theories tended to concentrate on superpositions of repetitive motions, but by the 1970s ideas of chaos theory began to dominate. And in fact the widespread assumption emerged that between randomness in the environment, quantum randomness and chaos theory almost any observed randomness in nature could be accounted for. Traditional mathematical models of natural systems are often expressed in terms of probabilities, but do not normally involve anything one can explicitly consider as randomness. Models used in computer simulations, however, do very often use explicit randomness. For not knowing about the phenomenon of intrinsic randomness generation, it has normally been assumed that with the kinds of discrete elements and fairly simple rules common in such models, realistically complicated behavior can only ever be obtained if explicit randomness is continually introduced.

■ **Applications of randomness.** See page 1192.

■ **Sources of randomness.** Two simple mechanical methods for generating randomness seem to have been used in almost every civilization throughout recorded history. One is to toss an object and see which way up or where it lands; the other is to select an object from a collection mixed by shaking. The first method has been common in games of chance, with polyhedral dice already existing in 2750 BC. The second—often called drawing lots—has normally been used when there is more at stake. It is mentioned several times in the Bible, and even today remains the most common method for large lotteries. (See page 969.) Variants include methods such as drawing straws. In antiquity fortune-telling from randomness often involved looking say at growth patterns of goat entrails or sheep shoulder blades; today configurations of tea leaves are sometimes considered. In early modern times the matching of fracture patterns in broken tally sticks was used to identify counterparties in financial contracts. Horse races and other events used as a basis for gambling can be viewed as randomness sources. Children's games like musical chairs in effect generate randomness by picking arbitrary stopping times. Games of chance based on wheels seem to have existed in Roman times; roulette developed in the 1700s. Card shuffling (see page 974) has been used as a source of randomness since at least the 1300s. Pegboards (as on page 312) were used to demonstrate effects of randomness in the late 1800s. An explicit table of 40,000 random digits was created in 1927 by Leonard Tippett from details of census data. And in 1938 further tables were generated by Ronald Fisher from digits of logarithms. Several tables based on physical processes were produced, with the RAND Corporation in 1955 publishing a table of a million random digits obtained from an electronic roulette wheel. Beginning in the 1950s, however, it became increasingly common to use pseudorandom generators whenever long sequences were needed—with linear feedback shift registers being most popular in standalone electronic devices, and linear congruential generators in programs (see page 974). There nevertheless continued to be occasional work done on mechanical sources of randomness for toys and games, and on physical electronic sources for cryptography systems (see page 969).

Randomness from the Environment

■ **Page 301 · Stochastic models.** The mechanism for randomness discussed in this section is the basis for so-called stochastic models now widely used in traditional science. Typically the idea of these models is to approximate those elements of a system about which one does not know much by random variables. (See also page 588.) In the early work along these lines done by James Clerk Maxwell and others in the 1880s, analytical formulas were usually worked out for the probabilities of different outcomes. But when electronic computers became available in the 1940s, the so-called Monte Carlo method became increasingly popular, in which instead explicit simulations are performed with different choices of random variables, and then statistical averages are found. Early uses of the Monte Carlo method were mostly in physics, particularly for studies of neutron diffusion and particle shower generation in high-energy collisions. But by the 1980s the Monte Carlo method had also become common in other fields, and was routinely used in studying for example message flows in communication networks and pricing processes in financial markets. (See also page 1192.)

■ **Page 301 · Ocean surfaces.** See page 1001.

■ **Page 302 · Random walks.** See page 328.

■ **Page 302 · Electronic noise.** Three types of noise are commonly observed in typical devices:

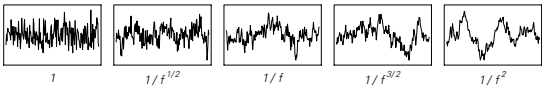
Shot noise. Electric currents are not continuous but are ultimately made up from large numbers of moving charge carriers, typically electrons. Shot noise arises from statistical fluctuations in the flow of charge carriers: if a single bit of data is represented by 10,000 electrons, the magnitude of the fluctuations will typically be about 1%. When looked at as a waveform over time, shot noise has a flat frequency spectrum.

Thermal (Johnson) noise. Even though an electric current may have a definite overall direction, the individual charge carriers within it will exhibit random motions. In a material

at nonzero temperature, the energy of these motions and thus the intensity of the thermal noise they produce is essentially proportional to temperature. (At very low temperatures, quantum mechanical fluctuations still yield random motion in most materials.) Like shot noise, thermal noise has a flat frequency spectrum.

Flicker (1/f) noise. Almost all electronic devices also exhibit a third kind of noise, whose main characteristic is that its spectrum is not flat, but instead goes roughly like $1/f$ over a wide range of frequencies. Such a spectrum implies the presence of large low-frequency fluctuations, and indeed fluctuations are often seen over timescales of many minutes or even hours. Unlike the types of noise described above, this kind of noise can be affected by details of the construction of individual devices. Although seen since the 1920s its origins remain somewhat mysterious (see below).

▪ **Power spectra.** Many random processes in nature show power spectra $Abs[Fourier[data]]^2$ with fairly simple forms. Most common are white noise uniform in frequency and $1/f^2$ noise associated with random walks. Other pure power laws $1/f^\alpha$ are also sometimes seen; the pictures below show some examples. (Note that the correlations in such data in some sense go like $t^{\alpha-1}$.) Particularly over the past few decades all sorts of examples of "$1/f$ noise" have been identified with $\alpha \simeq 1$, including flicker noise in resistors, semiconductor devices and vacuum tubes, as well as thunderstorms, earthquake and sunspot activity, heartbeat intervals, road traffic density and some DNA sequences. A pure $1/f^\alpha$ spectrum presumably reflects some form of underlying nesting or self-similarity, although exactly what has usually been difficult to determine. Mechanisms that generally seem able to give $\alpha \simeq 1$ include random walks with exponential waiting times, power-law distributions of step sizes (Lévy flights), or white noise variations of parameters, as well as random processes with exponentially distributed relaxation times (as from Boltzmann factors for uniformly distributed barrier heights), fractional integration of white noise, intermittency at transitions to chaos, and random substitution systems. (There was confusion in the late 1980s when theoretical studies of self-organized criticality failed correctly to take squares in computing power spectra.) Note that the Weierstrass function of page 918 yields a $1/f$ spectrum, and presumably suitable averages of spectra from any substitution system should also have $1/f^\alpha$ forms (compare page 586).



▪ **Page 303 · Spark chambers.** The sensitivity of sparks to microscopic details of the environment is highlighted by the several devices which essentially use them to detect the passage of individual elementary particles such as protons. Such particles leave a tiny trail of ionized gas, which becomes the path of the spark. This principle was used in Geiger counters, and later in spark chambers and wire chambers.

▪ **Physical randomness generators.** It is almost universally assumed that at some level physical processes must be the best potential sources of true randomness. But in practice their record has actually been very poor. It does not help that unlike algorithms physical devices can be affected by their environment, and can also not normally be copied identically. But in almost every case I know where detailed analysis has been done substantial deviations from perfect randomness have been found. This has however typically been attributed to engineering mistakes—or to sampling data too quickly—and not to anything more fundamental that is for example worth describing in publications.

▪ **Mechanical randomness.** It takes only small imperfections in dice or roulette wheels to get substantially non-random results (see page 971). Gaming regulations typically require dice to be perfect cubes to within one part in a few thousand; casinos normally retire dice after a few hundred rolls.

In processes like stirring and shaking it can take a long time for correlations to disappear—as in the phenomenon of long-time tails mentioned on page 999. One notable consequence were traces of insertion order among the 366 capsules used in the 1970 draft lottery in the U.S. But despite such problems mixing of objects remains by far the most common way to generate randomness when there is a desire for the public to see randomization occur. And so for example all the state lotteries in the U.S. are currently based on mixing between 10 and 54 balls. (Numbers games were instead sometimes based on digits of financial data in newspapers.)

There have been a steady stream of inventions for mechanical randomness generation. Some are essentially versions of dice. Others involve complicated cams or linkages, particularly for mechanical toys. And still others involve making objects like balls bounce around as randomly as possible in air or other fluids.

▪ **Electronic randomness.** Since the 1940s a steady stream of electronic devices for producing randomness have been invented, with no single one ever becoming widely used. An early example was the ERNIE machine from 1957 for British national lottery (premium bond) drawings, which worked by sampling shot noise from neon discharge

tubes—and perhaps because it extracted only a few digits per second no deviations from randomness in its output were found. (U.S. missiles apparently used a similar method to produce randomly spaced radar pulses for determining altitude.) Since the 1970s electronic randomness generators have typically been based on features of semiconductor devices—sometimes thermal noise, but more often breakdown, often in back-biased zener diodes. All sorts of schemes have been invented for getting unbiased output from such systems, and acceptable randomness can often be obtained at kilohertz rates, but obvious correlations almost always appear at higher rates. Macroscopic thermal diffusion undoubtedly underestimates the time for good microscopic randomization. For in addition to *1/f* noise effects, solitons and other collective lattice effects presumably lead to power-law decay of correlations. It still seems likely however that some general inequalities should exist between the rate and quality of randomness that can be extracted from a system with particular thermodynamic properties.

■ **Quantum randomness.** It is usually assumed that even if all else fails a quantum process such as radioactive decay will yield perfect randomness. But in practice the most accurate measurements show phenomena such as *1/f* noise, presumably as a result of features of the detector and perhaps of electromagnetic fields associated with decay products. Acceptable randomness has however been obtained at rates of tens of bits per second. Recent attempts have also been made to produce quantum randomness at megahertz rates by detecting paths of single photons. (See also page 1064.)

■ **Randomness in computer systems.** Most randomness needed in practical computer systems is generated purely by programs, as discussed on page 317. But to avoid having a particular program give exactly the same random sequence every time it is run, one usually starts from a seed chosen on the basis of some random feature of the environment. Until the early 1990s this seed was most often taken from the exact time of day indicated by the computer's clock at the moment when it was requested. But particularly in environments where multiple programs can start almost simultaneously other approaches became necessary. Versions of the Unix operating system, for example, began to support a virtual device (typically called /dev/random) to maintain a kind of pool of randomness based on details of the computer system. Most often this uses precise timings between interrupts generated by keys being pressed, a mouse being moved, or data being delivered from a disk, network, or other device. And to prevent the same state being reached every time a computer is rebooted, some information is permanently maintained in a file. At the end of the 1990s standard microprocessors also began to include instructions to sample thermal noise from an on-chip resistor. (Any password or encryption key made up by a human can be thought of as a source of randomness; some systems look at details of biometric data, or scribbles drawn with a mouse.)

■ **Randomness in biology.** Thermal fluctuations in chemical reactions lead to many kinds of microscopic randomness in biological systems, sometimes amplified when organisms grow. For example, small-scale randomness in embryos can affect large-scale pigmentation patterns in adult organisms, as discussed on page 1013. Random changes in single DNA molecules can have global effects on the development of an organism. Standard mitotic cell division normally produces identical copies of DNA—with random errors potentially leading for example to cancers. But in sexual reproduction genetic material is rearranged in ways normally assumed by classical genetics to be perfectly random. One reason is that which sperm fertilizes a given egg is determined by random details of sperm and fluid motion. Another reason is that egg and sperm cells get half the genetic material of an organism, somewhat at random. In most cells, say in humans, there are two versions of all 23 chromosomes—one from the father and one from the mother. But when meiosis forms egg and sperm cells they get only one version of each. There is also exchange of DNA between paternal and maternal chromosomes, typically with a few crossovers per chromosome, at positions that seem more or less randomly distributed among many possibilities (the details affect regions of repeating DNA used for example in DNA fingerprinting).

In the immune system blocks of DNA—and joins between them—are selected at random by microscopic chemical processes when antibodies are formed.

Most animal behavior is ultimately controlled by electrical activity in nerve cells—and this can be affected by details of sensory input, as well as by microscopic chemical processes in individual cells and synapses (see page 1011).

Flagellated microorganisms can show random changes in direction as a result of tumbling when their flagella counter-rotate and the filaments in them flail around.

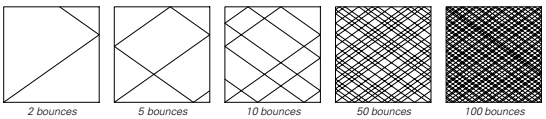
(See also page 1011.)

Chaos Theory and Randomness from Initial Conditions

■ **Page 305 · Spinning and tossing.** Starting with speed v, the speed of the ball at time t is simply $v - at$, where a is the deceleration produced by friction. The ball thus stops at time

v/a. The distance gone by the ball at a given time is $x = v\,t - a\,t^2/2$, and its orientation is $Mod[x, 2\,\pi\,r]$. For dice and coins there are some additional detailed effects associated with the shapes of these objects and the way they bounce. (Polyhedral dice have become more common since Dungeons & Dragons became popular in the late 1970s.) Note that in practice a coin tossed in the air will typically turn over between ten and twenty times while a die rolled on a table will turn over a few tens of times. A coin spun on a table can rotate several hundred times before falling over and coming to rest.

▪ **Billiards.** A somewhat related system is formed by a billiard ball bouncing around on a table. The issue of which sequence of horizontal and vertical sides the ball hits depends on the exact slope with which the ball is started (in the picture below it is $1/\sqrt{2}$). In general, it is given by the successive terms in the continued fraction form (see page 914) of this slope, and is related to substitution systems (see page 903). (See also page 1022.)



▪ **Fluttering.** If one releases a stationary piece of paper in air, then unlike a coin, it does not typically maintain the same orientation as it falls. Small pieces of paper spin in a repetitive way; but larger pieces of paper tend to flutter in a seemingly random way (as discussed, among others, by James Clerk Maxwell in 1853). A similar phenomenon can be seen if one drops a coin in water. I suspect that in these cases the randomness that occurs has an intrinsic origin, rather than being the result of sensitive dependence on initial conditions.

▪ **History of chaos theory.** The idea that small causes can sometimes have large effects has been noted by historians and others since antiquity, and captured for example in "for want of a nail … a kingdom was lost". In 1860 James Clerk Maxwell discussed how collisions between hard sphere molecules could lead to progressive amplification of small changes and yield microscopic randomness in gases. In the 1870s Maxwell also suggested that mechanical instability and amplification of infinitely small changes at occasional critical points might explain apparent free will (see page 1135). (It was already fairly well understood that for example small changes could determine which way a beam would buckle.) In 1890 Henri Poincaré found sensitive dependence on initial conditions in a particular case of the three-body problem (see below), and later proposed that such phenomena could be common, say in meteorology. In 1898 Jacques Hadamard noted general divergence of trajectories in spaces of negative curvature, and Pierre Duhem discussed the possible general significance of this in 1908. In the 1800s there had been work on nonlinear oscillators—particularly in connection with models of musical instruments—and in 1927 Balthazar van der Pol noted occasional "noisy" behavior in a vacuum tube oscillator circuit presumably governed by a simple nonlinear differential equation. By the 1930s the field of dynamical systems theory had begun to provide characterizations of possible forms of behavior in differential equations. And in the early 1940s Mary Cartwright and John Littlewood noted that van der Pol's equation could exhibit solutions somehow sensitive to all digits in its initial conditions. The iterated map $x \to 4\,x\,(1 - x)$ was also known to have a similar property (see page 918). But most investigations centered on simple and usually repetitive behavior—with any strange behavior implicitly assumed to be infinitely unlikely. In 1962, however, Edward Lorenz did a computer simulation of a set of simplified differential equations for fluid convection (see page 998) in which he saw complicated behavior that seemed to depend sensitively on initial conditions—in a way that he suggested was like the map $x \to FractionalPart[2\,x]$. In the mid-1960s, notably through the work of Steve Smale, proofs were given that there could be differential equations in which such sensitivity is generic. In the late 1960s there began to be all sorts of simulations of differential equations with complicated behavior, first mainly on analog computers, and later on digital computers. Then in the mid-1970s, particularly following discussion by Robert May, studies of iterated maps with sensitive dependence on initial conditions became common. Work by Robert Shaw in the late 1970s clarified connections between information content of initial conditions and apparent randomness of behavior. The term "chaos" had been used since antiquity to describe various forms of randomness, but in the late 1970s it became specifically tied to the phenomenon of sensitive dependence on initial conditions. By the early 1980s at least indirect signs of chaos in this sense (see note below) had been seen in all sorts of mechanical, electrical, fluid and other systems, and there emerged a widespread conviction that such chaos must be the source of all important randomness in nature. So in 1985 when I raised the possibility that intrinsic randomness might instead be a key phenomenon this was greeted with much hostility by some younger proponents of chaos theory. Insofar as what they had to say was of a scientific nature, their main point was that somehow what I

had seen in cellular automata must be specific to discrete systems, and would not occur in the continuous systems assumed to be relevant in nature. But from many results in this book it is now clear that this is not correct. (Note that James Gleick's 1987 popular book *Chaos* covers somewhat more than is usually considered chaos theory—including some of my results on cellular automata from the early 1980s.)

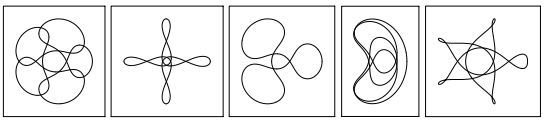
■ **Information content of initial conditions.** See page 920.

■ **Recognizing chaos.** Any system that depends sensitively on digits in its initial conditions must necessarily be able to show behavior that is not purely repetitive (compare page 955). And when it is said that chaos has been found in a particular system in nature what this most often actually means is just that behavior with no specific repetition frequency has been seen (compare page 586). To give evidence that this is not merely a reflection of continual injection of randomness from the environment what is normally done is to show that at least some aspect of the behavior of the system can be fit by a definite simple iterated map or differential equation. But inevitably the fit will only be approximate, so there will always be room for effects from randomness in the environment. And in general this kind of approach can never establish that sensitive dependence on initial conditions is actually the dominant source of randomness in a given system—say as opposed to intrinsic randomness generation. (Attempts are sometimes made to detect sensitive dependence directly by watching whether a system can do different things after it appears to return to almost exactly the same state. But the problem is that it is hard to be sure that the system really is in the same state—and that there are not all sorts of large differences that do not happen to have been observed.)

■ **Instability.** Sensitive dependence on initial conditions is associated with a kind of uniform instability in systems. But vastly more common in practice is instability only at specific critical points—say bifurcation points—combined with either intrinsic randomness generation or randomness from the environment. (Note that despite its widespread use in discussions of chaos theory, this is also what usually seems to happen with the weather; see page 1177.)

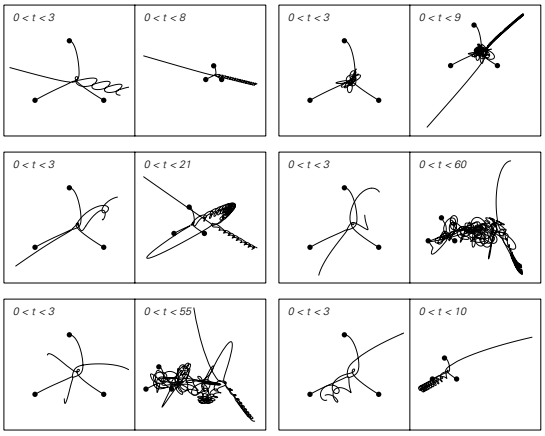
■ **Page 313 · Three-body problem.** The two-body problem was analyzed by Johannes Kepler in 1609 and solved by Isaac Newton in 1687. The three-body problem was a central topic in mathematical physics from the mid-1700s until the early 1900s. Various exact results were obtained—notably the existence of stable equilateral triangle configurations corresponding to so-called Lagrange points. Many approximate practical calculations, particularly on the Earth-Moon-Sun system, were done using series expansions involving thousands of algebraic terms. (It is now possible to get most results just by direct numerical computation using for example *NDSolve*.) From its basic setup the three-body system conserves standard mechanical quantities like energy and angular momentum. But it was thought it might also conserve other quantities (or so-called integrals of the motion). In 1887, however, Heinrich Bruns showed that there could be no such quantities expressible as algebraic functions of the positions and velocities of the bodies (in standard Cartesian coordinates). In the mid-1890s Henri Poincaré then showed that there could also be no such quantities analytic in positions, velocities and mass ratios. And from these results the conclusion was drawn that the three-body problem could not be solved in terms of algebraic formulas and integrals. In 1912 Karl Sundman did however find an infinite series that could in principle be summed to give the solution—but which converges exceptionally slowly. And even now it remains conceivable that the three-body problem could be solved in terms of more sophisticated standard mathematical functions. But I strongly suspect that in fact nothing like this will ever be possible and that instead the three-body problem will turn out to show the phenomenon of computational irreducibility discussed in Chapter 12 (and that for example three-body systems are universal and in effect able to perform any computation). (See also page 1132.)

In Henri Poincaré's study of the collection of possible trajectories for three-body systems he identified sensitive dependence on initial conditions (see above), noted the general complexity of what could happen (particularly in connection with so-called homoclinic tangles), and developed topology to provide a simpler overall description. With appropriate initial conditions one can get various forms of simple behavior. The pictures below show some of the possible repetitive orbits of an idealized planet moving in the plane of a pair of stars that are in a perfect elliptical orbit.



The pictures below show results for a fairly typical sequence of initial conditions where all three bodies interact. (The two bodies at the bottom are initially at rest; the body at the top is given progressively larger rightward velocities.) What generically happens is that one of the bodies escapes from the other two (like t or sometimes $t^{2/3}$). Often this happens quickly, but sometimes all three bodies show complex and

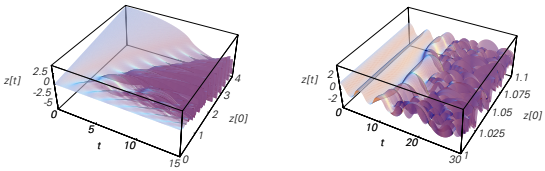
apparently random behavior for quite a while. (The delay before escaping is reminiscent of resonant scattering.)



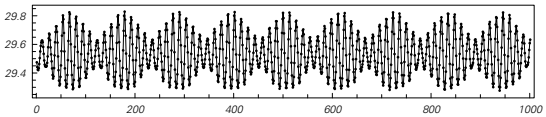
▪ **Page 314 · Simple case.** The position of the idealized planet in the case shown satisfies the differential equation

$$\partial_{tt} z[t] == -z[t]/(z[t]^2 + (1/2\,(1 + e\,\mathrm{Sin}[2\,\pi\,t]))^2)^{3/2}$$

where e is the eccentricity of the elliptical orbit of the stars ($e = 0.1$ in the picture). (Note that the physical situation is unstable: if the planet is perturbed so that there is a difference between its distance to each star, this will tend to increase.) Except when $e = 0$, the equation has no solution in terms of standard mathematical functions. It can be solved numerically in *Mathematica* using *NDSolve*, although a working precision of 40 decimal digits was used to obtain the results shown. Following work by Kirill Sitnikov in 1960 and by Vladimir Alekseev in 1968, it was established that with suitably chosen initial conditions, the equation yields any sequence *Floor[t[i + 1] - t[i]]* of successive zero-crossing times *t[i]*. The pictures below show the dependence of *z[t]* on *t* and *z[0]*. As *t* increases, *z[t]* typically begins to vary more rapidly with *z[0]*—reflecting sensitive dependence on initial conditions.



▪ **Page 314 · Randomness in the solar system.** Most motion observed in the solar system on human timescales is highly regular—though sometimes intricate, as in the sequence of numbers of days between successive new moons shown below. In the mid-1980s, however, work by Jack Wisdom and others established that randomness associated with sensitive dependence on initial conditions could occur in certain current situations in the solar system, notably in the orbits of asteroids. Various calculations suggest that there should also be sensitive dependence on initial conditions in the orbits of planets in the solar system—with effects doubling every few million years. But there are so far no observational signs of randomness resulting from this, and indeed the planets—at least now—mostly just seem to have orbits that are within a few percent of circles. If a planet moved in too random a way then it would tend to collide or escape from the solar system. And indeed it seems quite likely that in the past there may have been significantly more planets in our solar system—with only those that maintained regular orbits now being left. (See also page 1021.)



The Intrinsic Generation of Randomness

▪ **Autoplectic processes.** In the 1985 paper where I introduced intrinsic randomness generation I called processes that show this autoplectic, while I called processes that transcribe randomness from outside homoplectic.

▪ **Page 316 · Algorithmic randomness.** The idea of there being no simple procedure that can generate a particular sequence can be stated more precisely by saying that there is no program shorter than the sequence itself which can be used to generate the sequence, as discussed in more detail on page 1067.

▪ **Page 317 · Randomness in *Mathematica*.** *SeedRandom[n]* is the function that sets up the initial conditions for the cellular automaton. The idea of using this kind of system in general and this system in particular as a source of randomness was described in my 1987 U.S. patent number 4,691,291.

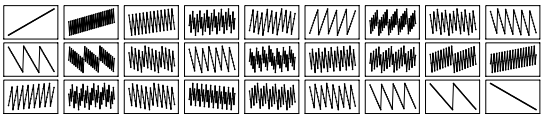
▪ **Page 321 · Cellular automata.** From the discussion here it should not be thought that in general there is necessarily anything better about generating randomness with cellular automata than with systems based on numbers. But the point is that the specific method used for making practical linear congruential generators does not yield particularly good randomness and has led to some incorrect intuition about the generation of randomness. If one goes beyond the specifics of linear congruential generators, then one can find many features of systems based on numbers that seem to be

perfectly random, as discussed in Chapter 4. In addition, one should recognize that while the complete evolution of the cellular automaton may effectively generate perfect randomness, there may be deviations from randomness introduced when one constructs a practical random number generator with a limited number of cells. Nevertheless, no such deviations have so far been found except when one looks at sequences whose lengths are close to the repetition period. (See however page 603.)

▪ **Page 321 · Card shuffling.** Another rather poor example of intrinsic randomness generation is perfect card shuffling. In a typical case, one splits the deck of cards in two, then carefully riffles the cards so as to make alternate cards come from each part of the deck. Surprisingly enough, this simple procedure, which can be represented by the function

```
s[list_] := Flatten[
    Transpose[Reverse[Partition[list, Length[list]/2]]]]
```

with or without the *Reverse*, is able to produce orderings which at least in some respects seem quite random. But by doing *Nest[s, Range[52], 26]* one ends up with a simple reversal of the original deck, as in the pictures below.



▪ **Random number generators.** A fairly small number of different types of random number generators have been used in practice, so it is possible to describe all the major ones here.

Linear congruential generators. The original suggestion made by Derrick Lehmer in 1948 was to take a number n and at each step to replace it by *Mod[a n, m]*. Lehmer used $a = 23$ and $m = 10^8 + 1$. Most subsequent implementations have used $m = 2^j$, often with $j = 31$. Such choices are particularly convenient on computers where machine integers are represented by 32 binary digits. The behavior of the linear congruential generator depends greatly on the exact choice of a. Starting with the so-called RANDU generator used on mainframe computers in the 1960s, a common choice made was $a = 65539$. But as shown in the main text, this choice leads to embarrassingly obvious regularities. Starting in the mid-1970s, another common choice was $a = 69069$. This was also found to lead to regularities, but only in six or more dimensions. (Small values of a also lead to an excess of runs of identical digits, as mentioned on page 903.)

The repetition period for a generator with rule $n \to$ *Mod[a n, m]* is given (for a and m relatively prime) by *MultiplicativeOrder[a, m]*. If m is of the form 2^j, this implies a maximum period for any a of $m/4$, achieved when *MemberQ[{3, 5}, Mod[a, 8]]*. In general the maximum period is *CarmichaelLambda[m]*, where the value $m - 1$ can be achieved for prime m.

As illustrated in the main text, when $m = 2^j$ the right-hand base 2 digits in numbers produced by linear congruential generators repeat with short periods; a digit k positions from the right will typically repeat with period no more than 2^k. When $m = 2^j - 1$ is prime, however, even the rightmost digit repeats only with period $m - 1$ for many values of a.

More general linear congruential generators use the basic rule $n \to$ *Mod[a n + b, m]*, and in this case, $n = 0$ is no longer special, and a repetition period of exactly m can be achieved with appropriate choices of a, b and m. Note that if the period is equal to its absolute maximum of m, then every possible n is always visited, whatever n one starts from. Page 962 showed diagrams that represent the evolution for all possible starting values of n.

Each point in the 2D plots in the main text has coordinates of the form *{n[i], n[i + 1]}* where *n[i + 1] = Mod[a n[i], m]*. If one could ignore the *Mod*, then the coordinates would simply be *{n[i], a n[i]}*, so the points would lie on a single straight line with slope a. But the presence of the *Mod* takes the points off this line whenever $a\,n[i] \geq m$. Nevertheless, if a is small, there are long runs of *n[i]* for which the *Mod* is never important. And that is why in the case $a = 3$ the points in the plot fall on obvious lines.

In the case $a = 65539$, the points lie on planes in 3D. The reason for this is that

```
n[i + 2] == Mod[65539^2 n[i], 2^31] ==
  Mod[6 n[i + 1] - 9 n[i], 2^31]
```

so that in computing *n[i + 2]* from *n[i + 1]* and *n[i]* only small coefficients are involved.

It is a general result related to finding short vectors in lattices that for some d the quantity *n[i + d]* can always be written in terms of the *n[i + k]*'s; $k < d$ using only small coefficients. And as a consequence, the points produced by any linear congruential generator must lie on regular hyperplanes in some number of dimensions.

(For cryptanalysis of linear congruential generators see page 1089.)

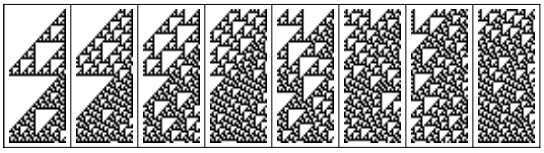
Linear feedback shift registers. Used since the 1950s, particularly in special-purpose electronic devices, these systems are effectively based on running additive cellular automata such as rule 60 in registers with a limited number

of cells and with a certain type of spiral boundary conditions. In a typical case, each cell is updated using

```
LFSRStep[list_] :=
  Append[Rest[list], Mod[list[[1]] + list[[2]], 2]]
```

with a step of cellular automaton evolution corresponding to the result of updating all cells in the register. As with additive cellular automata, the behavior obtained depends greatly on the length n of the register. The maximal repetition period of $2^n - 1$ can be achieved only if `Factor[1 + x + x^n, Modulus → 2]` finds no factors. (For $n < 512$, this is true when $n = 1, 2, 3, 4, 6, 7, 9, 15, 22, 28, 30, 46, 60, 63, 127, 153, 172, 303$ or 471. Maximal period is assured when in addition `PrimeQ[2^n - 1]`.) The pictures below show the evolution obtained for $n = 30$ with

```
NestList[Nest[LFSRStep, #, n] &,
  Append[Table[0, {n - 1}], 1], t]
```



Like additive cellular automata as discussed on page 951, states in a linear feedback shift register can be represented by a polynomial `FromDigits[list, x]`. Starting from a single 1, the state after t steps is then given by

```
PolynomialMod[x^t, {1 + x + x^n, 2}]
```

This result illustrates the analogy with linear congruential generators. And if the distribution of points generated is studied with the Cantor set geometry, the same kind of problems occur as in the linear congruential case (compare page 1094).

In general, linear feedback shift registers can have "taps" at any list of positions on the register, so that their evolution is given by

```
LFSRStep[taps_List, list_] :=
  Append[Rest[list], Mod[Apply[Plus, list[[taps]]], 2]]
```

(With taps specified by the positions of 1's in a vector of 0's, the inside of the `Mod` can be replaced by `vec . list` as on page 1087.) For a register of size n the maximal period of $2^n - 1$ is obtained whenever $x^n + $ `Apply[Plus, x^(taps-1)]` is one of the `EulerPhi[2^n - 1]/n` primitive polynomials that appear in `Factor[Cyclotomic[2^n - 1, x], Modulus → 2]`. (See pages 963 and 1084.)

One can also consider nonlinear feedback shift registers, as discussed on page 1088.

Generalized Fibonacci generators. It was suggested in the late 1950s that the Fibonacci sequence `f[n_] := f[n - 1] + f[n - 2]` modulo 2^k might be used with different choices of `f[0]` and `f[1]` as a random number generator (see page 891). This particular idea did not work well, but generalizations based on the recurrence `f[n_] := Mod[f[n - p] + f[n - q], 2^k]` have been studied extensively, for example with $p = 24$, $q = 55$. Such generators are directly related to linear feedback shift registers, since with a list of length q, each step is simply

```
Append[Rest[list], Mod[list[[1]] + list[[q - p + 1]], 2^k]]
```

Cryptographic generators. As discussed on page 598, so-called stream cipher cryptographic systems work essentially by generating a repeatable random sequence. Practical stream cipher systems can thus be used as random number generators. Starting in the 1980s, the most common example has been the Data Encryption Standard (DES) introduced by the U.S. government (see page 1085). Unless special-purpose hardware is used, however, this method has not usually been efficient enough for practical random number generation applications.

Quadratic congruential generators. Several generalizations of linear congruential generators have been considered in which nonlinear functions of n are used at each step. In fact, the first known generator for digital computers was John von Neumann's "middle square method"

```
n → FromDigits[Take[IntegerDigits[n^2, 10, 20], {5, 15}], 10]
```

In practice this generator has too short a repetition period to be useful. But in the early 1980s studies of public key cryptographic systems based on number theoretical problems led to some reinvestigation of quadratic congruential generators. The simplest example uses the rule

```
n → Mod[n^2, m]
```

It was shown that for $m = p\,q$ with p and q prime the sequence `Mod[n, 2]` was in a sense as difficult to predict as the number m is to factor (see page 1090). But in practice, the period of the generator in such cases is usually too short to be useful. In addition, there has been the practical problem that if n is stored on a computer as a 32-bit number, then n^2 can be 64 bits long, and so cannot be stored in the same way. In general, the period divides `CarmichaelLambda[CarmichaelLambda[m]]`. When m is a prime, this implies that the period can then be as long as $(m - 3)/2$. The largest m less than 2^{16} for which this is true is 65063, and the sequence generated in this case appears to be fairly random.

Cellular automaton generators. I invented the rule 30 cellular automaton random number generator in 1985. Since that time the generator has become quite widely used for a variety of applications. Essentially all the other generators discussed here have certain linearity properties which

allow for fairly complete analysis using traditional mathematical methods. Rule 30 has no such properties. Empirical studies, however, suggest that the repetition period, for example, is about $2^{0.63\,n}$, where n is the number of cells (see page 260). Note that rule 45 can be used as an alternative to rule 30. It has a somewhat longer period, but does not mix up nearby initial conditions as quickly as rule 30. (See also page 603.)

■ **Unequal probabilities.** Given a sequence a of n equally probable 0's and 1's, the following generates a single 0 or 1 with probabilities approximating $\{1-p, p\}$ to n digits:

```
Fold[{BitAnd, BitOr}[[1 + First[#2]]][#1, Last[#2]] &, 0,
  Reverse[Transpose[{First[RealDigits[p, 2, n, -1]], a}]]]
```

This can be generalized to allow a whole sequence to be generated with as little as an average of two input digits being used for each output digit.

■ **Page 323 · Sources of repeatable randomness.** In using repeatability to test for intrinsic randomness generation, one must avoid systems in which there is essentially some kind of static randomness in the environment. Sources of this include the profile of a rough solid surface, or the detailed patterns of grains inside a solid.

■ **Page 324 · Probabilistic rules.** There appears to be a discrete transition as a function of the size of the perturbations, similar to phase transitions seen in the phenomenon of directed percolation. Note that if one just uses the original cellular automata rules, then with any nonzero probability of reversing the colors of cells, the patterns will be essentially destroyed. With more complicated cellular automaton rules, one can get behavior closer to the continuous cellular automata shown here. (See also page 591.)

■ **Page 325 · Noisy cellular automata.** In correspondence with electronics, the continuous cellular automata used here can be thought of as analog models for digital cellular automata. The specific form of the continuous generalization of the modulo 2 function used is

```
λ[x_] := Exp[-10 (x - 1)^2] + Exp[-10 (x - 3)^2]
```

Each cell in the system is then updated according to $\lambda[a + c]$ for rule 90, and $\lambda[a + b + c + b\,c]$ for rule 30. Perturbations of size δ are then added using $v + \text{Sign}[v - 1/2]\,\text{Random}[]\,\delta$.

Note that the basic approach used here can be extended to allow discrete cellular automata to be approximated by partial differential equations where not only color but also space and time are continuous. (Compare page 464.)

■ **Page 326 · Repeatably random experiments.** Over the years, I have asked many experimental scientists about repeatability in seemingly random data, and in almost all cases they have told me that they have never looked for such a thing. But in a few cases they say that in fact on thinking about it they remember various forms of repeatability.

Examples where I have seen evidence of repeatable randomness as a function of time in published experimental data include temperature differences in thermal convection in closed cells of liquid helium, reaction rates in oxidation of carbon monoxide on catalytic surfaces, and output voltages from firings of excited single nerve cells. Typically there are quite long periods of time where the behavior is rather accurately repeatable—even though it may wiggle tens or hundreds in a seemingly random way—interspersed with jumps of some kind. In most cases the only credible models seem to be ones based on intrinsic randomness generation. But insofar as there is any definite model, it is inevitable that looking in sufficient detail at sufficiently many components of the system will reveal regularities associated with the underlying mechanism.

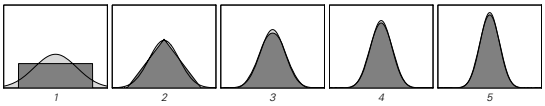
The Phenomenon of Continuity

■ **Discreteness in computer programs.** The reason for discreteness in computer programs is that the only real way we know how to construct such programs is using discrete logical structures. The data that is manipulated by programs can be continuous, as can the elements of their rules. But at some level one always gives discrete symbolic descriptions of the logical structure of programs. And it is then certainly more consistent to make both data and programs involve only discrete elements. In Chapter 12 I will argue that this approach is not only convenient, but also necessary if we are to represent our computations using processes that can actually occur in nature.

■ **Central Limit Theorem.** Averages of large collections of random numbers tend to follow a Gaussian or normal distribution in which the probability of getting value x is

```
Exp[-(x - μ)^2/(2 σ^2)]/(Sqrt[2 π] σ)
```

The mean μ and standard deviation σ are determined by properties of the random numbers, but the form of the distribution is always the same. The only conditions are that the random numbers should be statistically independent, and that their distribution should have bounded variance, so that, for example, the probability for very large numbers is rapidly damped. (The limit of an infinite collection of numbers gives $\sigma \to 0$ in accordance with the law of large numbers.) The pictures at the top of the next page show how averages of successively larger collections of uniformly distributed numbers converge to a Gaussian distribution.



The Central Limit Theorem leads to a self-similarity property for the Gaussian distribution: if one takes n numbers that follow Gaussian distributions, then their average should also follow a Gaussian distribution, though with a standard deviation that is $1/\sqrt{n}$ times smaller.

▪ **History.** That averages of random numbers follow bell-shaped distributions was known in the late 1600s. The formula for the Gaussian distribution was derived by Abraham de Moivre around 1733 in connection with theoretical studies of gambling. In the late 1700s Pierre-Simon Laplace did this again to predict the distribution of comet orbits, and showed that the same results would be obtained for other underlying distributions. Carl Friedrich Gauss made connections to the distribution of observational errors, and the relevance of the Gaussian distribution to biological and social systems was noted. Progressively more general proofs of the Central Limit Theorem were given from the early 1800s to the 1930s. Many natural systems were found to exhibit Gaussian distributions—a typical example being height distributions for humans. (Weight distributions are however closer to lognormal; compare page 1003.) And when statistical methods such as analysis of variance became established in the early 1900s it became increasingly common to assume underlying Gaussian distributions. (Gaussian distributions were also found in statistical mechanics in the late 1800s.)

▪ **Related results.** Gaussian distributions arise when large numbers of random variables get added together. If instead such variables (say probabilities) get multiplied together what arises is the lognormal distribution

```
Exp[-(Log[x] - μ)^2/(2 σ^2)]/(Sqrt[2 π] x σ)
```

For a wide range of underlying distributions the extreme values in large collections of random variables follow the Fisher-Tippett distribution

```
Exp[(x - μ)/β] Exp[-Exp[(x - μ)/β]]/β
```

related to the Weibull distribution used in reliability analysis.

For large symmetric matrices with random entries following a distribution with mean 0 and bounded variance the density of normalized eigenvalues tends to Wigner's semicircle law

```
2 Sqrt[1 - x^2] UnitStep[1 - x^2]/π
```

while the distribution of spacings between tends to

```
1/2 (π x) Exp[1/4 (-π) x^2]
```

The distribution of largest eigenvalues can often be expressed in terms of Painlevé functions.

(See also $1/f$ noise on page 969.)

▪ **Page 328 · Random walks.** In one dimension, a random walk with t steps of length 1 starting at position 0 can be generated from

```
NestList[# + (-1)^Random[Integer] &, 0, t]
```

or equivalently

```
FoldList[Plus, 0, Table[(-1)^Random[Integer], {t}]]
```

A generalization to d dimensions is then

```
FoldList[Plus, Table[0, {d}], Table[RotateLeft[PadLeft[
  {(-1)^Random[Integer]}, d], Random[Integer, d - 1]], {t}]]
```

A fundamental property of random walks is that after t steps the root mean square displacement from the starting position is proportional to $\sqrt{t}$. In general, the probability distribution for the displacement of a particle that executes a random walk is

```
With[{σ = 1}, (d/(2 π σ t))^(d/2) Exp[-d r^2/(2 σ t)]]
```

The same results are obtained, with a different value of σ, for other random microscopic rules, so long as the variance of the distribution of step lengths is bounded (as in the Central Limit Theorem).

As mentioned on page 1082, the frequency spectrum `Abs[Fourier[list]]^2` for a 1D random walk goes like $1/\omega^2$.

The character of random walks changes somewhat in different numbers of dimensions. For example, in 1D and 2D, there is probability 1 that a particle will eventually return to its starting point. But in 3D, this probability (on a simple cubic lattice) drops to about 0.341, and in d dimensions the probability falls roughly like `1/(2 d)`. After a large number of steps t, the number of distinct positions visited will be proportional to t, at least above 2 dimensions (in 2D, it is proportional to `t/Log[t]` and in 1D $\sqrt{t}$). Note that the outer boundaries of patterns like those on page 330 formed by n random walks tend to become rougher when t is much larger than `Log[n]`.

To make a random walk on a lattice with k directions in two dimensions, one can set up

```
e = Table[{Cos[2 π s/k], Sin[2 π s/k]}, {s, 0, k - 1}]
```

then use

```
FoldList[Plus, {0, 0}, Table[e[[Random[Integer, {1, k}]]], {t}]]
```

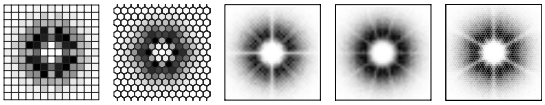
It turns out that on any regular lattice, in any number of dimensions, the average behavior of a random walk is always isotropic. As discussed in the note below, this can be viewed as a consequence of the fact that the probability distribution in a random walk depends only on

```
Sum[Outer[Times, e[[s]], e[[s]]], {s, Length[e]}]
```

and not on products of more of the `e[[s]]`.

There are nevertheless some properties of random walks that are not isotropic. The picture below, for example, shows the

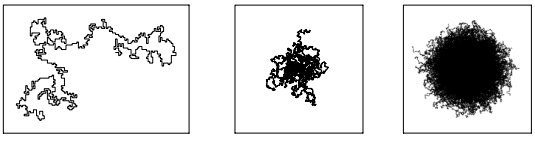
so-called extreme value distribution of positions furthest from the origin reached after 10 steps and 100 steps by random walks on various lattices.



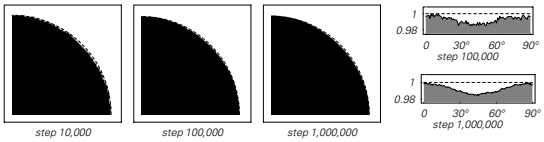
In the pictures in the main text, all particles start out at a particular position, and progressively spread out from there. But in general, one can consider sources that emit new particles every step, or absorbers and reflectors of particles. The average distribution of particles is given in general by the diffusion equation shown on page 163. The solutions to this equation are always smooth and continuous.

A physical example of an approximation to a random walk is the spreading of ink on blotting paper.

■ **Self-avoiding walks.** Any walk where the probabilities for a given step depend only on a fixed number of preceding steps gives the same kind of limiting Gaussian distribution. But imposing the constraint that a walk must always avoid anywhere it has been before (as for example in an idealized polymer molecule) leads to correlations over arbitrary times. If one adds individual steps at random then in 2D one typically gets stuck after perhaps a few tens of steps. But tricks are known for generating long self-avoiding walks by combining shorter walks or successively pivoting pieces starting with a simple line. The pictures below show some 1000-step examples. They look in many ways similar to ordinary random walks, but their limiting distribution is no longer strictly Gaussian, and their root mean square displacement after t steps varies like $t^{3/4}$. (In $d \le 4$ dimensions the exponent is close to the Flory mean field theory value $3/(2+d)$; for $d > 4$ the results are the same as without self-avoidance.)



■ **Page 331 · Basic aggregation model.** This model appears to have first been described by Murray Eden in 1961 as a way of studying biological growth, and was simulated by him on a computer for clusters up to about 32,000 cells. By the mid-1980s clusters with a billion cells had been grown, and a very surprising slight anisotropy had been observed. The pictures below show which cells occur in more than 10% of 1000 randomly grown clusters. There is a 2% or so anisotropy that appears to remain essentially fixed for clusters above perhaps a million cells, tucking them in along the diagonal directions. The width of the region of roughness on the surface of each cluster varies with the radius of the cluster approximately like $r^{1/3}$. The most extensive use of the model in practice has been for studying tumor growth: currently a typical tumor at detection contains about a billion cells, and it is important to predict what protrusions there will be that can break off and form additional tumors elsewhere.



■ **Implementation.** One way to represent a cluster is by giving a list of the coordinates at which each black cell occurs. Then starting with a single black cell at the origin, represented by *{{0, 0}}*, the cluster can be grown for t steps as follows:

```
AEvolve[t_] := Nest[AStep, {{0, 0}}, t]

AStep[c_] := (If[! MemberQ[c, #], Append[c, #],
     AStep[c]] &)[f[c] + f[{{1, 0}, {0, 1}, {-1, 0}, {0, -1}}]]
f[a_] := a[[Random[Integer, {1, Length[a]}]]]
```

This implementation can easily be extended to any type of lattice and any number of dimensions. Even with various additional optimizations, it is remarkable how much slower it is to grow a cluster with a model that requires external random input than to generate similar patterns with models such as cellular automata that intrinsically generate their own randomness.

The implementation above is a so-called type B Eden model in which one first selects a cell in the cluster, then randomly selects one of its neighbors. One gets extremely similar results with a type A Eden model in which one just randomly selects a cell from all the ones adjacent to the cluster. With a grid of cells set up in advance, each step in this type of Eden model can be achieved with

```
AStep[a_] := ReplacePart[a, 1, (#[[Random[
     Integer, {1, Length[#]}]]] &)[Position[(1 - a) Sign[
     ListConvolve[{{0, 1, 0}, {1, 0, 1}, {0, 1, 0}}, a, {2, 2}]], 1]]]
```

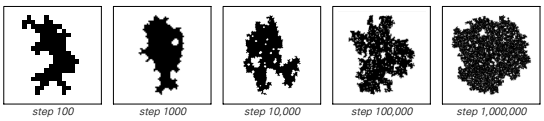
This implementation can readily be extended to generalized aggregation models (see below).

■ **Page 332 · Generalized aggregation models.** One can in general have rules in which new cells can be added only at positions whose neighborhoods match specific templates (compare page 213). There are 32 possible symmetric such rules with just 4 immediate neighbors—of which 16 lead to growth (from any seed), and all seem to yield at least

approximately circular clusters (of varying densities). Without symmetry, all sorts of shapes can be obtained, as in the pictures below. (The rule numbers here follow the scheme on page 927 with offsets *{{-1, 0}, {0, -1}, {0, 1}, {1, 0}}*). Note that even though the underlying rule involves randomness definite geometrical shapes can be produced. An extreme case is rule 2, where only a single neighborhood with a single black cell is allowed, so that growth occurs along a single line.

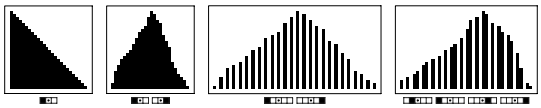
If one puts conditions on where cells can be added one can in principle get clusters where no further growth is possible. This does not seem to happen for rules that involve 4 neighbors, but with 8 neighbors there are cases in which clusters can get fairly large, but end up having no sites where further cells can be added. The pictures below show examples for a rule that allows growth except when there are exactly 1, 3 or 4 neighbors (totalistic constraint 242).

The question of what ultimate forms of behavior can occur with any sequence of random choices, starting from a given configuration with a given rule, is presumably in general undecidable. (It has some immediate relations to tiling problems and to halting problems for non-deterministic Turing machines.) With the rule illustrated above, however, those clusters that do successfully grow exhibit complicated and irregular shapes, but nevertheless eventually seem to take on a roughly circular shape, as in the pictures below.

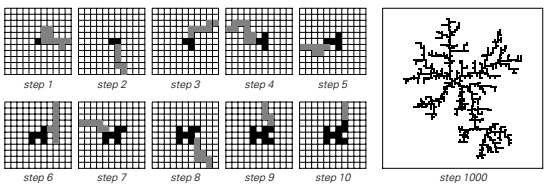


At some level the basic aggregation model of page 331 has a deterministic outcome: after sufficiently many steps every cell will be black. But most generalized aggregation models do not have this property: instead, the form of their internal patterns depends on the sequence of random choices made. Particularly with more than two colors it is however possible to arrange that the internal pattern always ends up being the same, or at least has patches that are the same—essentially by using rules with the confluence property discussed on page 1036.

The pictures below show 1D generalized aggregation systems with various templates. The second one is the analog of the system from page 331.

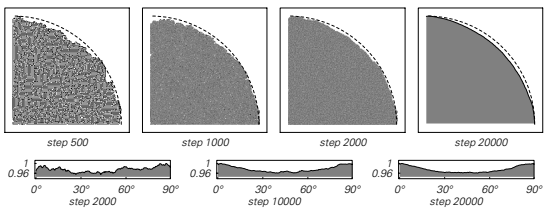


▪ **Page 333 · Diffusion-limited aggregation (DLA).** While many 2D cellular automata produce intricate nested shapes, the aggregation models shown here seem to tend to simple limiting shapes. Most likely there are some generalized aggregation models for which this is not the case. And indeed this phenomenon has been seen in other systems with randomness in their underlying rules. An example studied extensively in the 1980s is diffusion-limited aggregation (DLA). The idea of this model is to add cells to a cluster one at a time, and to determine where a cell will be added by seeing where a random walk that starts far from the cluster first lands on a square adjacent to the cluster. An example of the behavior obtained in this model is shown below:

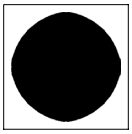


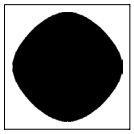
The lack of smooth overall behavior in this case can perhaps be attributed to the global probing of the cluster that is effectively done by each incoming random walk. (See also page 994.)

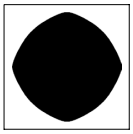
▪ **Page 334 · Code 746.** Much as in the aggregation model above, the pictures below show that there is a slight deviation from perfect circular growth, with an anisotropy that appears to remain roughly fixed at perhaps 4% above a few thousand steps (corresponding to patterns with a few million cells).



■ **Other rules.** The pictures below show patterns generated after 10,000 steps with several rules, starting respectively from rows of 7, 6, 7 and 11 cells (compare pages 177 and 181). The outer boundaries are somewhat smooth, though definitely not circular. In the second rule shown, the interior of the pattern always continues to change; in the others it remains essentially fixed.


code 746


code 29408


code 175850


code 174826

■ **Isotropy.** Any pattern grown from a single cell according to rules that do not distinguish different directions on a lattice must show the same symmetry as the lattice. But we have seen that in fact many rules actually yield almost circular patterns with much higher symmetry. One can characterize the symmetry of a pattern by taking the list v of positions of cells it contains, and looking at tensors of successive ranks n:

```
Apply[Plus,
  Map[Apply[Outer[Times, ##] &, Table[#, {n}]] &, v]]
```

For circular or spherical patterns that are perfectly isotropic in d dimensions these tensors must all be proportional to

```
(d - 2)!! Array[Apply[Times, Map[(1 - Mod[#, 2]) (# - 1)!! &,
  Table[Count[{##}, i], {i, d}]]] &, Table[d, {n}]]/(d + n - 2)!!
```

For odd n this is inevitably true for any lattice with mirror symmetry. But for even n it can fail. For a square lattice, it still nevertheless always holds up to $n = 2$ (so that the analogs of moments of inertia satisfy $\mathcal{I}_{xx} == \mathcal{I}_{yy}$, $\mathcal{I}_{xy} == \mathcal{I}_{yx} == 0$). And for a hexagonal lattice it holds up to $n = 4$. But when $n = 4$ isotropy requires the $\{1, 1, 1, 1\}$ and $\{1, 1, 2, 2\}$ tensor components to have ratio $\beta = 3$—while square symmetry allows these components to have any ratio. (In general there will be more than one component unless the representation of the lattice symmetry group carried by the rank n tensor is irreducible.) In 3D no regular lattice forces isotropy beyond $n = 2$, while in 4D the SO(8) lattice works up to $n = 4$, in 8D the E_8 lattice up to $n = 6$, and in 24D the Leech lattice up to $n = 10$. (Lattices that give dense sphere packings tend to show more isotropy.) Note that isotropy can also be characterized using analogs of multipole moments, obtained in 2D by summing $r_i \, \text{Exp}[i \, n \, \theta_i]$, and in higher dimensions by summing appropriate *SphericalHarmonicY* or *GegenbauerC* functions. For isotropy, only the $n = 0$ moment can be nonzero. On a 2D lattice with m directions, all moments are forced to be zero except when m divides n. (Sums of squares of moments of given order in general provide rotationally invariant measures of anisotropy—equal to pair correlations weighted with *LegendreP* or *GegenbauerC* functions.)

Even though it is not inevitable from lattice symmetry, one might think that if there is some kind of effective randomness in the underlying rules then sufficiently large patterns would still often show some sort of average isotropy. And at least in the case of ordinary random walks, they do, so that for example, the ratio averaged over all possible walks of $n = 4$ tensor components after t steps on a square lattice is $\beta = 3 + 2/(t - 1)$, converging to the isotropic value 3, and the ratio of $n = 6$ components is $5 - 4/(t - 1) + 32/(3\,t - 4)$. For the aggregation model of page 331, β also decreases with t, reaching 4 around $t = 10$, but now its asymptotic value is around 3.07.

In continuous systems such as partial differential equations, isotropy requires that coordinates in effect appear only in ∇. In most finite difference approximations, there is presumably isotropy in the end, but the rates of convergence are almost inevitably rather different in different directions relative to the lattice.

■ **Page 336 · Domains.** Some of the effective rules for interfaces between black and white domains are easy to state. Given a flat interface, the layer of cells immediately on either side of this interface behaves like the rule 150 1D cellular automaton. On an infinitely long interface, protrusions of cells with one color into a domain of the opposite color get progressively smaller, eventually leaving only a certain pattern of cells in the layer immediately on one side of the interface. 90° corners in an otherwise flat interface effectively act like reflective boundary conditions for the layer of cells on top of the interface.

The phenomenon of domains illustrated here is also found in various 2D cellular automata with 4-neighbor rather than 8-neighbor rules. One example is totalistic code 52, which is a direct analog in the 4-neighbor case of the rule illustrated here. Other examples are outer totalistic codes 111, 293, 295 and 920. The domain boundaries in these cases, however, are not as clear as for the 8-neighbor totalistic rule with code 976 that is shown here.

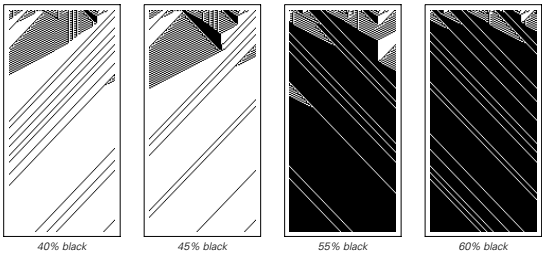
■ **Spinodal decomposition.** The separation into progressively larger black and white regions seen in the cellular automata shown here is reminiscent of the phenomena that occur for example in the separation of randomly mixed oil and water. Various continuous models of such processes have been proposed, notably the Cahn-Hilliard equation from 1958. One feature often found is that the average radius of "droplets" increases with time roughly like $t^{1/3}$.

Origins of Discreteness

■ **Page 339 · 1D transitions.** There are no examples of the phenomenon shown here among the 256 rules with two possible colors and depending only on nearest neighbors. Among the 4,294,967,296 rules that depend on next-nearest neighbors, there are a handful of examples, including rules with numbers 4196304428, 4262364716, 4268278316 and 4266296876. The behavior obtained with the first of these rules is shown below. An example that depends on three neighbors on each side was discovered by Peter Gacs, Georgii Kurdyumov and Leonid Levin in 1978, following work on how reliable electronic circuits can be built from unreliable components by Andrei Toom:

```
{a1_, a2_, a3_, a4_, a5_, a6_, a7_} →
  If[If[a4 == 1, a1 + a3 + a4, a4 + a5 + a7] ≥ 2, 1, 0]
```

The 4-color rule shown in the text is probably the clearest example available in one dimension. It has rule number 2948697645239957498148900977948124938 24.



■ **Page 340 · 2D transitions.** The simplest symmetrical rules (such as 4-neighbor totalistic code 56) which make the new color of a cell be the same as the majority of the cells in its neighborhood do not exhibit the discrete transition phenomenon, but instead lead to fixed regions of black and white. The 4-neighbor rule with totalistic code 52 can be used as an alternative to the second rule shown here. A probabilistic version of the first rule shown here was discussed by Andrei Toom in 1980.

■ **Phase transitions.** The discrete transitions shown in cellular automata in this section are examples of general phenomena known in physics as phase transitions. A phase transition can be defined as any discontinuous change that occurs in a system with a large number of components when a parameter associated with that system is varied. (Some physicists might argue for a somewhat narrower definition that allows only discontinuities in the so-called partition function of equilibrium statistical mechanics, but for many of the most interesting applications, the definition I use is the appropriate one.) Standard examples of phase transitions include boiling, melting, sublimation (solids such as dry ice turning into gases), loss of magnetization when a ferromagnet is heated, alignment of molecules in liquid crystals above a certain electric field (the basis for liquid crystal displays), and the onset of superconductivity and superfluidity at low temperatures.

It is conventional to distinguish two kinds of phase transitions, often called first-order and higher-order. First-order transitions occur when a system has two possible states, such as liquid and gas, and as a parameter is varied, which of these states is the stable one changes. Boiling and melting are both examples of first-order transitions, as is the phenomenon shown in the cellular automaton in the main text. Note that one feature of first-order transitions is that as soon as the transition is passed, the whole system always switches completely from one state to the other.

Higher-order transitions are in a sense more gradual. On one side of the transition, a system is typically completely disordered. But when the transition is passed, the system does not immediately become completely ordered. Instead, its order increases gradually from zero as the parameter is varied. Typically the presence of order is signalled by the breaking of some kind of symmetry—say of rotational symmetry by the spontaneous selection of a preferred direction.

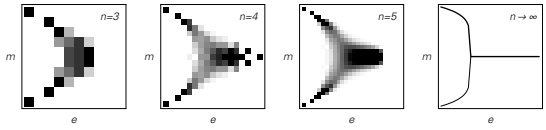
■ **The Ising model.** The 2D Ising model is a prototypical example of a system with a higher-order phase transition. Introduced by Wilhelm Lenz in 1920 as an idealization of ferromagnetic materials (and studied by Ernst Ising) it involves a square array *s* of spins, each either up or down (+1 or -1), corresponding to two orientations for magnetic moments of atoms. The magnetic energy of the system is taken to be

```
e[s_] := -1/2 Apply[Plus, s ListConvolve[
          {{0, 1, 0}, {1, 0, 1}, {0, 1, 0}}, s, 2], {0, 1}]
```

so that each pair of adjacent spins contributes -1 when they are parallel and +1 when they are not. The overall magnetization of the system is given by `m[s_] := Apply[Plus, s, {0, 1}]`.

In physical ferromagnetic materials what is observed is that at high temperature, corresponding to high internal energy, there is no overall magnetization. But when the temperature goes below a critical value, spins tend to line up, and an overall magnetization spontaneously develops. In the context of the 2D Ising model this phenomenon is associated with the fact that those configurations of a large array of spins that have high total energy are overwhelmingly likely to have near zero overall magnetization, while those that have low

total energy are overwhelmingly likely to have nonzero overall magnetization. For an $n \times n$ array s of spins there are a total of 2^{n^2} possible configurations. The pictures below show the results of picking all configurations with a given energy $e[s]$ (cyclic boundary conditions are assumed) and then working out their distribution of magnetization values $m[s]$. Even for small n the pictures demonstrate that for large $e[s]$ the magnetization $m[s]$ is likely to be close to zero, but for smaller $e[s]$ two branches approaching +1 and -1 appear. In the limit $n \to \infty$ the distribution of magnetization values becomes sharp, and a definite discontinuous phase transition is observed.



Following the work of Lars Onsager around 1944, it turns out that an exact solution in terms of traditional mathematical functions can be found in this case. (This seems to be true only in 2D, and not in 3D or higher.) Almost all spin configurations with $e[s] > -\sqrt{2}$ (where here and below all quantities are divided by the total number of spins, so that $-2 \le e[s] \le 2$ and $-1 \le m[s] \le +1$) yield $m[s] == 0$. But for smaller $e[s]$ one can show that

```
Abs[m[s]] == (1 - Sinh[2 β]^-4)^1/8
```

where β can be deduced from

```
e[s] == -(Coth[2 β] (1 + 2 EllipticK[4 Sech[2 β]^2 Tanh[2 β]^2] (-1 + 2 Tanh[2 β]^2))/π))
```

This implies that just below the critical point $e_0 = -\sqrt{2}$ (which corresponds to $\beta = Log[1 + \sqrt{2}]/2$) $Abs[m] \sim (e_0 - e)^{1/8}$, where here 1/8 is a so-called critical exponent. (Another analytical result is that for $e \sim e_0$ correlations between pairs of spins can be expressed in terms of Painlevé functions.)

Despite its directness, the approach above of considering sets of configurations with specific energies $e[s]$ is not how the Ising model has usually been studied. Instead, what has normally been done is to take the array of spins to be in thermal equilibrium with a heat bath, so that, following standard statistical mechanics, each possible spin configuration occurs with probability $Exp[-\beta e[s]]$, where β is inverse temperature. It nevertheless turns out that in the limit $n \to \infty$ this so-called canonical ensemble approach yields the same results for most quantities as the microcanonical approach that I have used; β simply appears as a parameter, as in the formulas above.

About actual spin systems evolving in time the Ising model itself does not make any statement. But whenever the evolution is ergodic, so that all states of a given energy are visited with equal frequency, the average behavior obtained will at least eventually correspond to the average over all states discussed above.

In Monte Carlo studies of the Ising model one normally tries to sample states with appropriate probabilities by randomly flipping spins according to a procedure that can be thought of as emulating interaction with a heat bath. But in most actual physical spin systems it seems unlikely that there will be so much continual interaction with the environment. And from my discussion of intrinsic randomness generation it should come as no surprise that even a completely deterministic rule for the evolution of spins can make the system visit possible states in an effectively random way.

Among the simplest possible types of rules all those that conserve the energy $e[s]$ turn out to have behavior that is too simple and regular. And indeed, of the 4096 symmetric 5-neighbor rules, only identity and complement conserve $e[s]$. Of the 2^{32} general 5-neighbor rules 34 conserve $e[s]$—but all have only very simple behavior. (Compositions of several such rules can nevertheless yield complex behavior. Note that as indicated on page 1022, 34 of the 256 elementary 1D rules conserve the analog of $e[s]$.) Of the 262,144 9-neighbor outer totalistic rules the only ones that conserve $e[s]$ are identity and complement. But among all 2^{512} 9-neighbor rules, there are undoubtedly examples that show effectively random behavior. One marginally more complicated case effectively involving 13 neighbors is

```
IsingEvolve[list_, t_Integer] :=
  First[Nest[IsingStep, {list, Mask[list]}, t]]
IsingStep[{a_, mask_}] := {MapThread[
    If[#2 == 2 && #3 == 1, 1 - #1, #1] &, {a, ListConvolve[
      {{0, 1, 0}, {1, 0, 1}, {0, 1, 0}}, a, 2], mask}, 2], 1 - mask}
```

where

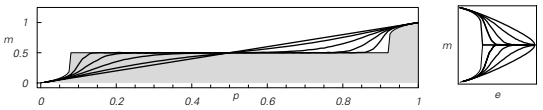
```
Mask[list_] := Array[Mod[#1 + #2, 2] &, Dimensions[list]]
```

is set up so that alternating checkerboards of cells are updated on successive steps.

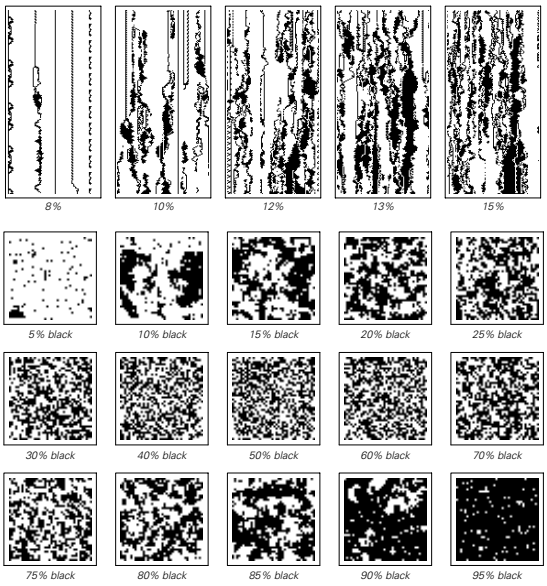
One can see a phase transition in this system by looking at the dependence of behavior on conserved total energy $e[s]$. If there are no correlations between spins, and a fraction p of them are +1, then $m[s] == p$ and $e[s] == -2(1 - 2p)^2$. And since the evolution conserves $e[s]$ changing the initial value of p allows one to sample different total energies. But since the evolution does not conserve $m[s]$ the average of this after many steps can be expected to be typical of all possible states of given $e[s]$.

The pictures at the top of the next page show the values of $m[s]$ (densities of +1 cells) after 0, 10, 100 and 1000 steps for a 500×500 system as a function of the initial values of $m[s]$ and $e[s]$. Also shown is the result expected for an infinite system at infinite time. (The slow approach to this limit can

be viewed as being a consequence of smallness of finite size scaling exponents in Ising-like systems.)



The phase transition in the Ising model is associated with a lack of smoothness in the dependence of the final m value on e or the initial value p of m in limiting cases of the pictures above. The transition occurs at $e = -\sqrt{2}$, corresponding to $p = (1 \pm 2^{-1/4})/2$. The pictures show typical configurations generated after 1000 steps from various initial densities, as well as slices through their evolution.



And what one sees at least roughly is that right around the phase transition there are patches of black and white of all sizes, forming an approximately nested random pattern. (See also pages 989 and 1149.)

▪ **General features of phase transitions.** To reproduce the Ising model, a cellular automaton must have several special properties. In addition to conserving energy, its evolution must be reversible in the sense discussed on page 435. And with the constraint of reversibility, it turns out that it is impossible to get a non-trivial phase transition in any 1D system with the kind of short-range interactions that exist in a cellular automaton. But in systems whose evolution is not reversible, it is possible for phase transitions to occur in 1D, as the examples in the main text show.

One point to notice is that the sharp change which characterizes any phase transition can only be a true discontinuity in the limit of an infinitely large system. In the case of the system on page 339, for example, it is possible to find special configurations with a finite total number of cells which lead to behavior opposite to what one expects purely on the basis of their initial density of black cells. When the total number of cells increases, however, the fraction of such configurations rapidly decreases, and in the infinite size limit, there are no such configurations, and a truly discontinuous transition occurs exactly at density 1/2.

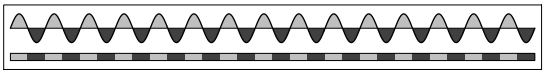
The discrete nature of phase transitions was at one time often explained as a consequence of changes in the symmetry of a system. The idea is that symmetry is either present or absent, and there is no continuous variation of level of symmetry possible. Thus, for example, above the transition, the Ising model treats up and down spins exactly the same. But below the transition, it effectively makes a choice of one spin direction or the other. Similarly, when a liquid freezes into a crystalline solid, it effectively makes a choice about the alignment of the crystal in space. But in boiling, as well as in a number of model examples, there is no obvious change of symmetry. And from studying phase transitions in cellular automata, it does not seem that an interpretation in terms of symmetry is particularly useful.

A common feature of phase transitions is that right at the transition point, there is competition between both phases, and some kind of nested structure is typically formed, as discussed on page 273 and above. The overall form and fractal dimension of this nested structure is typically independent of small-scale features of the system, making it fairly universal, and amenable to analysis using the renormalization group approach (see page 955).

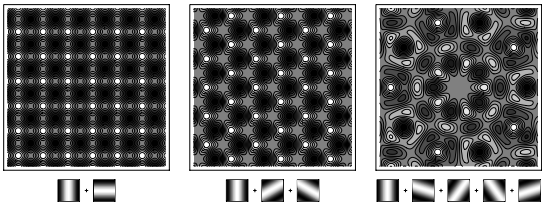
▪ **Percolation.** A simple example of a phase transition studied extensively since the 1950s involves taking a square lattice and filling in at random a certain density of black cells. In the limit of infinite size, there is a discrete transition at a density of about 0.592746, with zero probability below the transition to find a connected "percolating" cluster of black cells spanning the lattice, and unit probability above. (For a triangular lattice the critical density is exactly 1/2.) One can also study directed percolation in which one takes account of the connectivity of cells only in one direction on the lattice. (Compare the probabilistic cellular automata on pages 325 and 591. Note that the evolution of such systems is also analogous to the process of applying transfer matrices in studies of spin systems like Ising models.)

■ **Page 341 · Rate equations.** In standard chemical kinetics one assumes that molecules are uniformly distributed in space, so that the rates for particular reactions are proportional to the products of the densities of the molecules that react in them. Conditions for equilibrium where rates balance thus tend to be polynomial equations for densities—with discontinuous jumps in solutions sometimes occurring as parameters are changed. Analogous equations arise in probabilistic approximations to systems like cellular automata, as on page 953. But here—as well as in fast chemical reactions—correlations in spatial arrangements of elements tend to be important, invalidating simple probabilistic approaches. (For the cellular automaton on page 339 the simple condition for equilibrium is $p == p^2 (3 - 2 p)$, which correctly implies that 0, 1/2 and 1 are possible equilibrium densities.)

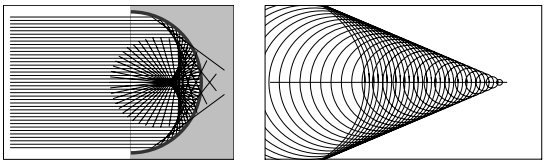
■ **Discreteness in space.** Many systems with continuous underlying rules generate discrete cellular structures in space. One common mechanism is for a wave of a definite wavelength to form (see page 988), and then for some feature of each cycle of this wave to be picked out, as in the picture below. In Chladni figures of sand on vibrating plates and in cloud streets in the atmosphere what happens is that material collects at points of zero displacement. And when a stream of water breaks up into discrete drops what happens is that oscillation minima yield necks that break.



Superpositions of waves at different angles can lead to various 2D cellular structures, as in the pictures below (compare page 1078).



Various forms of focusing and accumulation can also lead to discreteness in continuous systems. The first picture below shows a caustic or catastrophe in which a continuous distribution of light rays are focused by a circular reflector onto a discrete line with a cusp. The second picture shows a shock wave produced by an accumulation of circular waves emanating from a moving object—as seen in wakes of ships, sonic booms from supersonic aircraft, and Cerenkov light from fast-moving charged particles.



The Problem of Satisfying Constraints

■ **Rules versus constraints.** See page 940.

■ **NP completeness.** Finding 2D patterns that satisfy the constraints in the previous section is in general a so-called NP-complete problem. And this means that no known algorithm can be expected to solve this problem exactly for a size n array (say with given boundaries) in much less than 2^n steps (see page 1145). The same is true even if one allows a small fraction of squares to violate the constraints. However, the 1D version of the problem is not NP-complete, and in fact there is a specific rather efficient algorithm described on page 954 for solving it. Nevertheless, the procedures discussed in this section do not manage to make use of such specific algorithms, and in fact typically show little difference between problems that are and are not formally NP-complete.

■ **Page 343 · Distribution.** The distribution shown here rapidly approaches a Gaussian. (Note that in a 5×5 array, there are 10 interior squares that are subject to the constraints, while in a 10×10 array there are 65.) Very similar results seem to be obtained for constraints in a wide range of discrete systems.

■ **Page 346 · Implementation.** The number of squares violating the constraint used here is given by

```
Cost[list_] := Apply[Plus, Abs[list - RotateLeft[list]]]
```

When applied to all possible patterns, this function yields a distribution with Gaussian tails, but with a sharp point in the middle. Successive steps in the iterative procedure used on this page are given by

```
Move[list_] := (If[Cost[#] < Cost[list], #, list] &)[
    MapAt[1 - # &, list, Random[Integer, {1, Length[list]}]]]
```

while those in the procedure on page 347 have $\leq$ in place of $<$. The third curve shown on page 346 is obtained from

```
Table[Cost[IntegerDigits[i, 2, n]], {i, 0, 2^n - 1}]
```

There is no single ordering that makes all states which can be reached by changing a single square be adjacent. However, the ordering defined by *GrayCode* from page 901 does do this for one particular sequence of single square changes. The resulting curve is very similar to what is already shown.

■ **Page 347 · Iterative improvement.** The borders of the regions of black and white in the picture shown here essentially

follow random walks and annihilate in pairs so that their number decreases with time like $1/\sqrt{t}$. In 2D the regions are more complicated and there is no such simple behavior. Indeed starting from a particular state it is for example not clear whether it is ever possible to reach all other states.

■ **Gradient descent.** A standard method for finding a minimum in a smooth function `f[x]` is to use

```
FixedPoint[# - a f'[#] &, x0]
```

If there are local minima, then which one is reached will depend on the starting point x_0. It will not necessarily be the one closest to x_0 because of potentially complicated overshooting effects associated with the step size a. Newton's method for finding zeros of `f[x]` is related and is given by

```
FixedPoint[# - f[#]/f'[#] &, x0]
```

■ **Combinatorial optimization.** The problem of coming as close as possible to satisfying constraints in an arrangement of black and white squares is a simple example of a combinatorial optimization problem. In general, such problems involve minimization of a quantity that is determined by the arrangement of some set of discrete elements. A typical example is finding a placement of components in a 2D circuit so that the total length of wire necessary to the connect these components is minimized (related to the so-called travelling salesman problem). In using iterative procedures to solve combinatorial optimization problems, one issue is what kind of changes should be made at each step. In the main text we considered changing just one square at a time. But one can also change larger numbers of squares, or, for example, interchange whole blocks of squares. In general, the larger the changes made, the faster one can potentially approach a minimum, but the greater the chance is of overshooting. In the main text, we assumed that at each step we should always move closer to the minimum, or at least not get further away. But in trying to get over the kind of bumps shown in the third curve on page 346 it is sometimes better also to allow some probability of moving away from the minimum at a particular step. One approach is simulated annealing, in which one starts with this probability being large, and progressively decreases it. The notion is that at the beginning, one wants to move easily over the coarse features of a jagged curve, but then later home in on details. If the curve has a nested form, which appears to be the case in some combinatorial optimization problems, then this scheme can be expected to be at least somewhat effective. For the problems considered in the main text, simulated annealing provides some improvement but not much.

■ **Biologically motivated schemes.** The process of biological evolution by natural selection can be thought of as an iterative procedure for optimization. Usually, however, what is being optimized is some aspect of the form or behavior of an organism, which represents a very complicated constraint on the underlying genetic material. (It is as if one is defining constraints on the initial conditions for a cellular automaton by looking at the pattern generated by the cellular automaton after a long time.) But the strategies of biological evolution can also be used in trying to satisfy simpler constraints. Two of the most important strategies are maintaining a whole population of individuals, not just the single best result so far, and using sex to produce large-scale mixing. But once again, while these strategies may in some cases lead to greater efficiency, they do not usually lead to qualitative differences. (See also page 1105.)

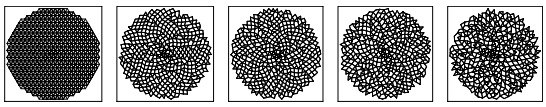
■ **History.** Work on combinatorial optimization started in earnest in the late 1950s, but by the time NP completeness was discovered in 1971 (see page 1143) it had become clear that finding exact solutions would be very difficult. Approximate methods tended to be constructed for specific problems. But in the early 1980s, simulated annealing was suggested by Scott Kirkpatrick and others as one of the first potentially general approaches. And starting in the mid-1980s, extensive work was done on biologically motivated so-called genetic algorithms, which had been advocated by John Holland since the 1960s. Progress in combinatorial optimization is however often difficult to recognize, because there are almost no general results, and results that are quoted are often sensitive to details of the problems studied and the computer implementations used.

■ **Page 349 · 2D cellular automata.** The rule numbers are specified as on page 927.

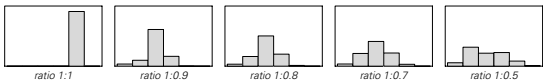
■ **Page 349 · Circle packings.** Hexagonal packing of equal circles has been known since early antiquity (e.g. the fourth picture on page 43). It fills a fraction $\pi/\sqrt{12} \simeq 0.91$ of area—which was proved maximal for periodic packings by Carl Friedrich Gauss in 1831 and for any packing by Axel Thue in 1910 and László Fejes Tóth in 1940. Much has been done to study densest packings of limited numbers of circles into various shapes, as well as onto surfaces of spheres (as in golf balls, pollen grains or radiolarians). Typically it has been found that with enough circles, patches of hexagonal packing always tend to form. (See page 987.)

For circles of unequal sizes rather little has been done. A procedure analogous to the one on page 350 was introduced by Charles Bennett in 1971 for 3D spheres (relevant for binary alloys). The picture below shows the

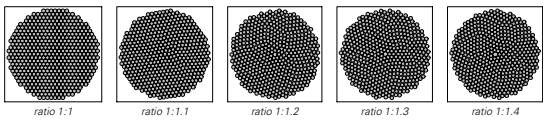
network of contacts between circles in the cases from page 350. Note that with the procedure used, each new circle added must immediately touch two existing ones, though subsequently it may get touched by varying numbers of other circles.



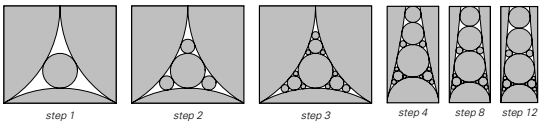
The distribution of numbers of circles that touch a given circle changes with the ratio of circle sizes, as in the picture below. The total filling fraction seems to vary fairly smoothly with this ratio, though I would not be surprised if some small-scale jumps were present.



Note that even a single circle of different size in the center can have a large-scale effect on the results of the procedure, as illustrated in the pictures below.



Finding densest packings of n circles is in general like solving quadratic programming problems with about n^2 constraints. But at least for many size ratios I suspect that the final result will simply involve each kind of circle forming a separated hexagonally-packed region. This will not happen, however, for size ratios $\leq 2/\sqrt{3} - 1 \simeq 0.15$, since then the small circles can fit into the interstices of an ordinary hexagonal pattern, yielding a filling fraction $1/18\,(17\sqrt{13} - 24)\,\pi \simeq 0.95$. The picture below shows what happens if one repeatedly inserts circles to form a so-called Apollonian packing derived from the problem studied by Apollonius of finding a circle that touches three others. At step t, 3^{t-1} circles are added for each original circle, and the network of tangencies among circles is exactly example (a) from page 509. Most of the circles added at a given step are not the same size, however, making the overall geometry not straightforwardly nested. (The total numbers of different sizes of circles for the first few steps are $\{2, 3, 5, 10, 24, 63, 178, 521\}$. At step 3, for example, the new circles have radii $(25 - 12\sqrt{3})/193$ and $(19 - 6\sqrt{3})/253$. In general, the radius of a circle inscribed between three other touching circles that have radii p, q, r is $p\,q\,r/(p\,q + p\,r + q\,r + 2\,\mathrm{Sqrt}[p\,q\,r\,(p + q + r)])$.) In the limit of an infinite number of steps the filling fraction tends to 1, while the region left unfilled has a fractal dimension of about 1.3057.



To achieve filling fraction 1 requires arbitrarily small circles, but there are many different arrangements of circles that will work, some not even close to nested. When actual granular materials are formed by crushing, there is probably some tendency to generate smaller pieces by following essentially substitution system rules, and the result may be a nested distribution of sizes that allows an Apollonian-like packing.

Apollonian packings turn out to correspond to limit sets invariant under groups of rational transformations in the complex plane. Note that as on page 1007 packings can be constructed in which the sizes of circles vary smoothly with position according to a harmonic function.

▪ **Sphere packings.** The 3D face-centered cubic (fcc) packing shown in the main text has presumably been known since antiquity, and has been used extensively for packing fruit, cannon balls, etc. It fills space with a density $\pi/\sqrt{18} \simeq 0.74$, which Johannes Kepler suggested in 1609 might be the maximum possible. This was proved for periodic packings by Carl Friedrich Gauss in 1831, and for any packing by Thomas Hales in 1998. (By offsetting successive layers hexagonal close packing (hcp) can be obtained; this has the same density as fcc, but has a trapezoid-rhombic dodecahedron Voronoi diagram—see note below and page 929—rather than an ordinary rhombic dodecahedron.)

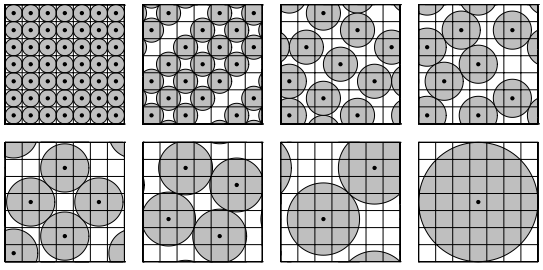
Random packings of spheres typically have densities around 0.64 (compared to 0.74 for fcc). Many of their large pores appear to be associated with poor packing of tetrahedral clusters of 4 spheres. (Note that individual such clusters—as well as for example 13-sphere approximate icosahedra—represent locally dense packings.)

It is common for shaking to cause granular materials (such as coffee or sand grains) to settle and pack at least a few percent better. Larger objects normally come to the top (as with mixed nuts, popcorn or pebbles and sand), essentially because the smaller ones more easily fall through interstices.

▪ **Higher dimensions.** In no dimension above 3 is it known for certain what configuration of spheres yields the densest packing. Cases in which spheres are arranged on repetitive lattices are related to error-correcting codes and groups. Up to 8D, the densest packings of this type are known to be ones obtained by successively adding layers individually

optimized in each dimension. And in fact up to 26D (with the exception of 11 through 13) all the densest packings known so far are lattices that work like this. In 8D and 24D these lattices are known to be ones in which each sphere touches the maximal number of others (240 and 196560 respectively). (In 8D the lattice also corresponds to the root vectors of the Lie group E_8; in 24D it is the Leech lattice derived from a Golay code, and related to the Monster Group). In various dimensions above 10 packings in which successive layers are shifted give slightly higher densities than known lattices. In all examples found so far the densest packings can always be repetitive; most can also be highly symmetrical—though in high dimensions random lattices often do not yield much worse results.

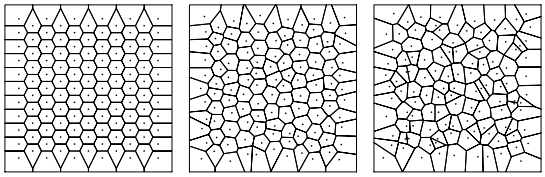
▪ **Discrete packings.** The pictures below show a discrete analog of circle packing in which one arranges as many circles as possible with a given diameter on a grid. (The grid is assumed to wrap around.)



The pictures show all the distinct maximal cases that exist for a 7×7 grid, corresponding to possible circles with diameters $Sqrt[m^2 + n^2]$. Already some of these are difficult to find. And in fact in general finding such packings is an NP-complete problem: it is equivalent to the problem of finding the maximum clique (completely connected set) in the graph whose vertices are joined whenever they correspond to grid points on which non-overlapping circles could be centered.

On large grids, optimal packings seem to approach rational approximations to hexagonal packings. But what happens if one generalizes to allow circles of different sizes is not clear.

▪ **Voronoi diagrams.** The Voronoi diagram for a set of points shows the region around each point in which one is closer to that point than to any other. (The edges of the regions are thus like watersheds.) The pictures below show a few examples. In 2D the regions in a Voronoi diagram are always polygons, and in 3D polyhedra. If all the points lie on a repetitive lattice each region will always be the same, and is often known as a Wigner-Seitz cell or a Dirichlet domain. For a simple cubic lattice the regions are cubes with 6 faces. For an fcc lattice they are rhombic dodecahedra with 12 faces and for a bcc lattice they are truncated octahedra (tetradecahedra) with 14 faces. (Compare page 929.)



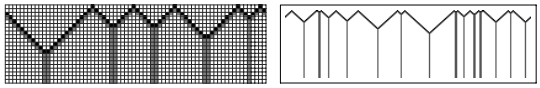
Voronoi diagrams for irregularly distributed points have found many applications. In 2D they are used in studies of animal territories, retail store utilization and municipal districting. In 3D they are used as simple models of foams, grains in solids, assemblies of biological cells and self-gravitating regions in primordial galaxy formation. Voronoi diagrams are relevant whenever there is growth in all directions at an identical speed from a collection of seed points. (In high dimensions they also appear immediately in studying error-correcting codes.)

Modern computational geometry has provided efficient algorithms for constructing Voronoi diagrams, and has allowed them to be used in mesh generation, point location, cluster analysis, machining plans and many other computational tasks.

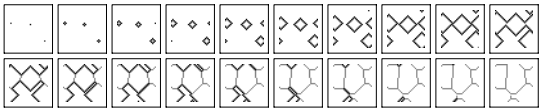
▪ **Discrete Voronoi diagrams.** The $k = 3$, $r = 1$ cellular automaton

$$\{\{0 \mid 1, n : (0 \mid 1), 0 \mid 1\} \to n, \{\_, 0, \_\} \to 2, \{\_, n\_, \_\} \to n - 1\}$$

is an example of a system that generates discrete 1D Voronoi diagrams by having regions that grow from every initial black cell, but stop whenever they meet, as shown below.



Analogous behavior can also be obtained in 2D, as shown for a 2D cellular automaton in the pictures below.



▪ **Brillouin zones.** A region in an ordinary Voronoi diagram shows where a given point is closest. One can also consider higher-order Voronoi diagrams in which each region shows where a given point is the k^{th} closest. The total area of each region is the same for every k, but some complexity in shape is seen, though for large k they always in a sense

approximate circles. 3D versions of such regions have been encountered in studies of quantum mechanical properties of crystals since the 1930s.

■ **Packing deformable objects.** If one pushes together identical deformable objects in 2D they tend to arrange themselves in a regular hexagonal array—and this configuration is known to minimize total boundary length. In 3D the arrangement one gets is typically not very regular—although as noted at various times since the 1600s individual objects often have pentagonal faces suggestive of dodecahedra. (The average number of faces for each object depends on the details of the random process used to pack them, but is typically around 14. Note that for a 3D Voronoi diagram with randomly placed points, the average number of faces for each region is $2 + 48\,\pi^2/35 \simeq 15.5$.) It was suggested by William Thomson (Kelvin) in 1887 that an array of 14-faced tetradecahedra on a bcc lattice might yield minimum total face area. But in 1993 Denis Weaire and Robert Phelan discovered a layered repetitive arrangement of 12- and 14-faced polyhedra (average 13.5) that yields 0.003 times less total area. It seems likely that there are polyhedra which fill space in a less regular way and yield still smaller total area. (Note that if the surfaces minimize area like soap films they are slightly curved in all these cases. See also pages 1007 and 1039.)

■ **Page 351 · Protein folding.** When the molecular structure of proteins was first studied in the 1950s it was assumed that given their amino acid sequences pure minimization of energy would determine their often elaborate overall shapes. But by the 1990s it was fairly clear that in fact many details of the actual processes by which proteins are assembled can greatly affect their specific pattern of folding. (Examples include effects of chaperone molecules and prions.) (See pages 1003 and 1184.)

Origins of Simple Behavior

■ **Previous approaches.** Before the discoveries in this book, nested and sometimes even repetitive behavior were quite often considered complex, and it was assumed that elaborate theories were necessary to explain them. Most of the theories that have been proposed are ultimately equivalent to what I discuss in this section, though they are usually presented in vastly more complicated ways.

■ **Uniformity in frequency.** As shown on page 587, a completely random sequence of cells yields a spectrum that is essentially uniform in frequency. Such uniformity in frequency is implied by standard quantum theory to exist in the idealized zero-point fluctuations of a free quantum field—with direct consequences for such semiclassical phenomena as the Casimir effect and Hawking radiation. (See page 1062.)

■ **Repetition in numbers.** A common source of repetition in systems involving numbers is the almost trivial fact that in a sequence of successive integers there is a repetitive pattern of cases at which a particular divisor occurs. Other examples include the repetitive structure of digits in rational numbers (see page 138) and continued fraction terms in square roots (see page 144).

■ **Repetition in continuous systems.** A standard approach to partial differential equations (PDEs) used for more than a century is so-called linear stability analysis, in which one assumes that small fluctuations around some kind of basic solution can be treated as a superposition of waves of the form $Exp[i\,k\,x]\,Exp[i\,\omega\,t]$. And at least in a linear approximation any given PDE then typically implies that ω is connected to the wavenumber k by a so-called dispersion relation, which often has a simple algebraic form. For some k this yields a value of ω that is real—corresponding to an ordinary wave that maintains the same amplitude. But for some k one often finds that ω has an imaginary part. The most common case $Im[\omega] > 0$ yields exponential damping. But particularly when the original PDE is nonlinear one often finds that $Im[\omega] < 0$ for some range of k—implying an instability which causes modes with certain spatial wavelengths to grow. The mode with the most negative $Im[\omega]$ will grow fastest, potentially leading to repetitive behavior that shows a particular dominant spatial wavelength. Repetitive patterns with this type of origin are seen in a number of situations, especially in fluids (and notably in connection with Kelvin-Helmholtz, Rayleigh-Taylor and other well-studied instabilities). Examples are ripples and swell on an ocean (compare page 1001), Bénard convection cells, cloud streets and splash coronas. Note that modes that grow exponentially inevitably soon become too large for a linear approximation—and when this approximation breaks down more complicated behavior with no sign of simple repetitive patterns is often seen.

■ **Examples of nesting.** Examples in which a single element splits into others include branching in plants, particle showers, genealogical trees, river deltas and crushing of rocks. Examples in which elements merge include river tributaries and some cracking phenomena.

■ **Page 358 · Nesting in numbers.** Chapter 4 contains several examples of systems based on numbers that exhibit nested behavior. Ultimately these examples can usually be traced to

nesting in the pattern of digits of successive integers, but significant translation is often required.

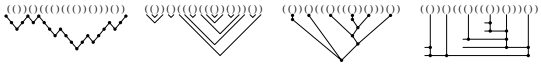
▪ **Nested lists.** One can think of structures that annihilate in pairs as being like parentheses or other delimiters that come in pairs, as in the picture below.

A string of balanced parentheses is analogous to a nested *Mathematica* list such as *{{{}, {{}}}, {}}*. The *Mathematica* expression tree for this list then has a structure analogous to the nested pattern in the picture.

The set of possible strings of balanced parentheses forms a context-free language, as discussed on page 939. The number of such strings containing $2n$ characters is the n^{th} Catalan number *Binomial[2 n, n]/(n + 1)* (as obtained from the generating function *(1 - Sqrt[1 - 4 x])/(2 x)*). The number of strings of depth d (and thus taking d steps to annihilate completely) is given by *c[{n, n}, d] - c[{n, n}, d - 1]* where

```
c[{_, _}, -1] = 0; c[{0, 0}, _] = 1; c[{m_, n_}, _] := 0 /; n > m;
c[{m_, n_}, d_] :=
  Sum[c[{i, j}, d], {i, 0, m - 1}, {j, m - d, n - 1}]
```

Several types of structures are equivalent to strings of balanced parentheses, as illustrated below.

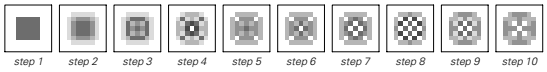


▪ **Phase transitions.** Nesting in systems like rule 184 (see page 273) is closely related to the phenomenon of scaling studied in phase transitions and critical phenomena since the 1960s. As discussed on page 983 ordinary equilibrium statistical mechanics effectively samples configurations of systems like rule 184 after large numbers of steps of evolution. But the point is that when the initial number of black and white cells is exactly equal—corresponding to a phase transition point—a typical configuration of rule 184 will contain domains with a nested distribution of sizes. The properties of such configurations can be studied by considering invariance under rescalings of the kind discussed on page 955, in analogy to renormalization group methods. A typical result is that correlations between colors of different cells fall off like a power of distance—with the specific power depending only on general features of the nested patterns formed, and not on most details of the system.

▪ **Self-organized criticality.** The fact that in traditional statistical mechanics nesting had been encountered only at the precise locations of phase transitions led in the 1980s to the notion that despite its ubiquity in nature nesting must somehow require fine tuning of parameters. Already in the early 1980s, however, my studies on simple additive and other cellular automata (see page 26) had for example made it rather clear that this is not the case. But in the late 1980s it became popular to think that in many systems nesting (as well as the largely unrelated phenomenon of $1/f$ noise) might be the result of fine tuning of parameters achieved through some automatic process of self-regulation. Computer experiments on various cellular automata and related systems were given as examples of how this might work. But in most of these experiments mistakes and misinterpretations were found, and in the end little of value was learned about the origins of nesting (or $1/f$ noise). Nevertheless, a number of interesting systems did emerge, the best known being the idealized sandpile model from the 1987 work of Per Bak, Chao Tang and Kurt Wiesenfeld. This is a $k = 8$ 2D cellular automaton in which toppling of sand above a critical slope is captured by updating an array of relative sand heights s according to the rule

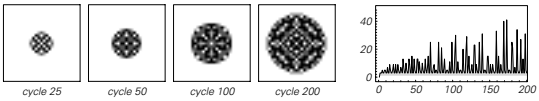
```
SandStep[s_] := s + ListConvolve[
    {{0, 1, 0}, {1, -4, 1}, {0, 1, 0}}, UnitStep[s - 4], 2, 0]
```

Starting from any initial condition, the rule eventually yields a fixed configuration with all values less than 4, as in the picture below. (With an $n \times n$ initial block of 4's, stabilization typically takes about $0.4\, n^2$ steps.).



To model the pouring of sand into a pile one can consider a series of cycles, in which at each cycle one first adds 4 to the value of the center cell, then repeatedly applies the rule until a new fixed configuration *FixedPoint[SandStep, s]* is obtained. (The more usual version of the model adds to a random cell.) The picture below shows slices through the evolution at several successive cycles. Avalanches of different sizes occur, yielding activity that lasts for varying numbers of steps.

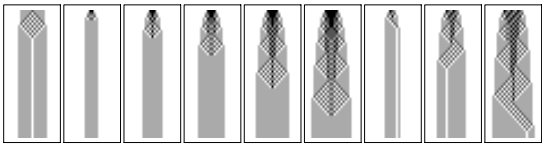
The pictures at the top of the next page show some of the final fixed configurations, together with the number of steps needed to reach them. (The total value of s at cycle t is $4t$; the radius of the nonzero region is about $0.74\sqrt{t}$.) The behavior one sees is fairly complicated—a fact which in the past resulted in much confusion and some bizarre claims, but which in the light of the discoveries in this book no longer seems surprising.



The system can be generalized to d dimensions as a $k = 4\,d$ cellular automaton with $2\,d$ final values. The total value of s is always conserved. In 1D, the update rule is simply

```
SandStep[s_] :=
  s + ListConvolve[{1, -2, 1}, UnitStep[s - 2], 2, 0]
```

In this case the evolution obtained if one repeatedly adds to the center cell (as in the first picture below) is always quite simple. But as the pictures below illustrate, evolution from typical initial conditions yields behavior that often looks a little like rule 184. With a total initial s value of m, the number of steps before a fixed point is reached seems to increase roughly like m^2.



When $d > 1$, more complicated behavior is seen for evolution from at least some initial conditions, as indicated above.

▪ **Random walks.** It is a consequence of the Central Limit Theorem that the pattern of any random walk with steps of bounded length (see page 977) must have a certain nested or self-similar structure, in the sense that rescaled averages of different numbers of steps will always yield patterns that look qualitatively the same. As emphasized by Benoit Mandelbrot in connection with a variety of systems in nature, the same is also true for random walks whose step lengths follow a power-law distribution, but are unbounded. (Compare page 969.)

▪ **Structure of algorithms.** The two most common overall frameworks that have traditionally been used in algorithms in computer science are iteration and recursion—and these correspond quite directly to having operations performed respectively in repetitive and nested ways. But while iteration is generally viewed as being quite easy to understand, until recently even recursion was usually considered rather difficult. No doubt the methods of this book will in the future lead to all sorts of algorithms based on much more complex patterns of behavior. (See page 1142.)

▪ **Origins of localized structures.** Much as with other features of behavior, one can identify several mechanisms that can lead to localized structures. In 1D, localized structures sometimes arise as defects in largely repetitive behavior, or more generally as boundaries between states with different properties—such as the different phases of the repetitive background in rule 110. In higher dimensions a common source—especially in systems that show some level of continuity—are point, line or other topological defects (see page 1045), of which vortices are a typical example.