EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

SECTION 7.9

Origins of Simple Behavior

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.

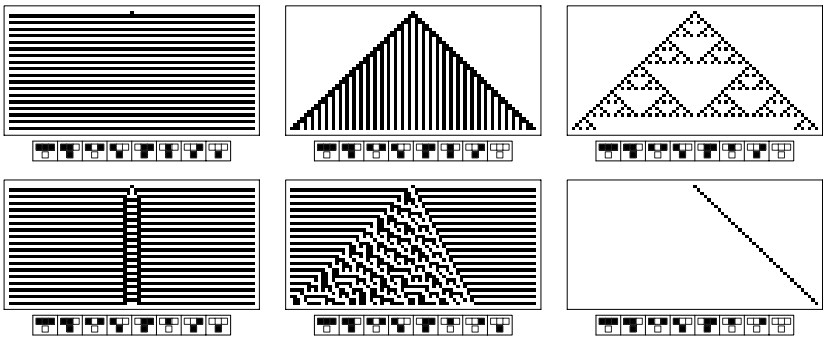
Origins of Simple Behavior

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.

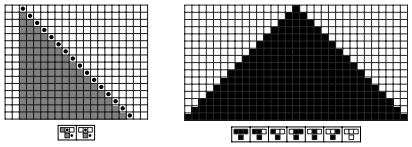
If one just looks at a rule in its raw form, it is usually almost impossible to tell much about the overall behavior it will produce. But in cases where this behavior ends up being simple, one can often recognize in it specific mechanisms that seem to be at work.

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.

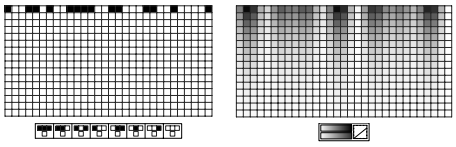
In many respects the very simplest possible type of behavior in any system is pure uniformity. And uniformity in time is particularly straightforward, for it corresponds just to no change occurring in the evolution of a system. But uniformity in space is already slightly more complicated, and indeed 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.



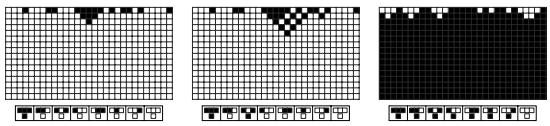
Homogenous growth from a single point is one straightforward way that uniformity in space can be produced, here illustrated in a mobile automaton and a cellular automaton.

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



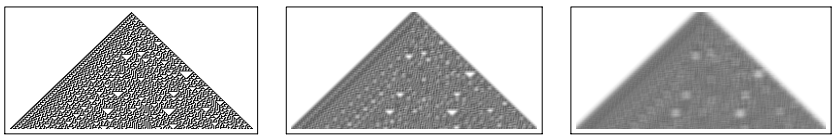
Uniformity in space can be achieved almost trivially if each element in a system independently evolves to the same state.

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.

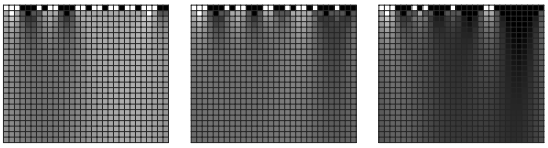


Averaging out small-scale randomness yields apparent uniformity, as shown here for a rule 30 pattern.

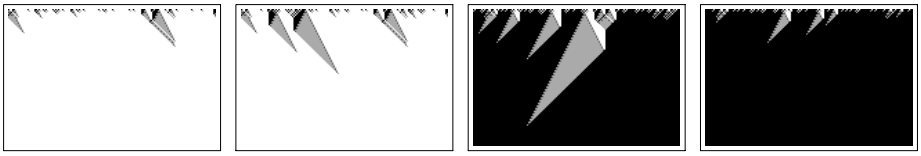
It is common for uniform behavior to be quite independent of initial conditions or other input to a system. But sometimes different uniform behavior can be obtained with different input.

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.

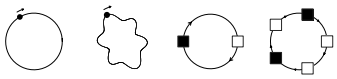


With different initial conditions this cellular automaton from page 339 can evolve either to uniform white or uniform black. Such discrete transitions are somewhat less common in one dimension than elsewhere.

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

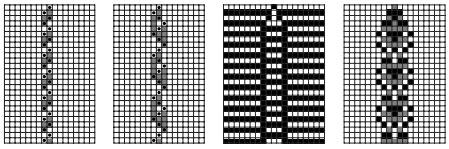


The behavior of a system will be repetitive in time whenever it effectively follows a closed curve—either literally in space, or in terms of states that it visits.

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

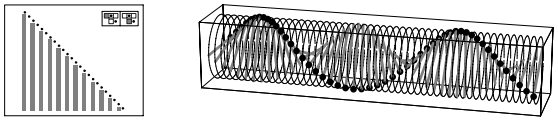
In some cases the basic structure of a system may allow only a limited number of possible states. But in other cases what happens is instead just that the actual evolution of a system never reaches more than a limited number of states.

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.



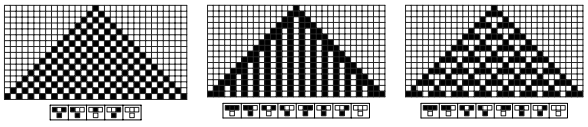
Examples of behavior in mobile automata and cellular automata that remains localized to a limited region and thus always eventually repeats.

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.

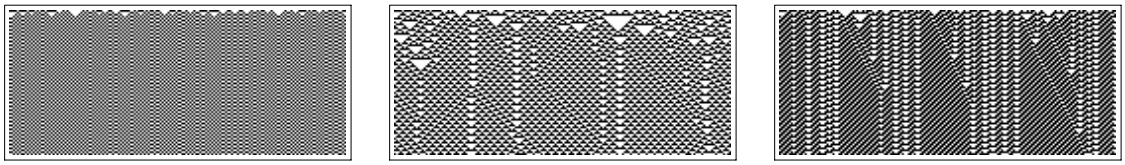
Growth from a simple seed can also readily lead to repetition in both space and time, as in the pictures below.



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

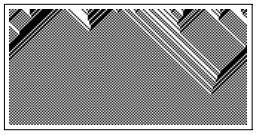
But what about random initial conditions? Repetition in time is still easy to achieve—say just by different parts of a system behaving independently. But repetition in space is slightly more difficult to achieve. For even if localized domains of repetition form, they need to have some mechanism for combining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

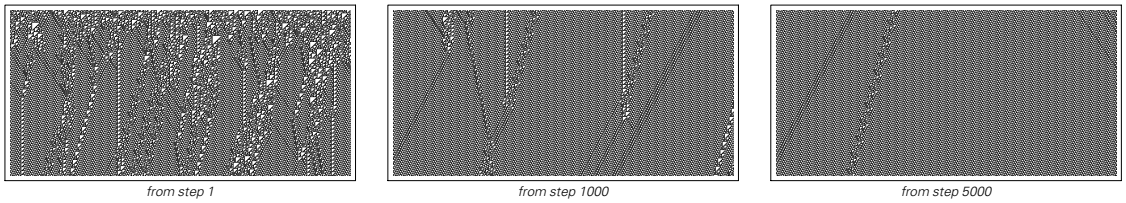
Cellular automata in which domains of repetitive behavior form, but in which walls typically remain forever between these domains.



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.



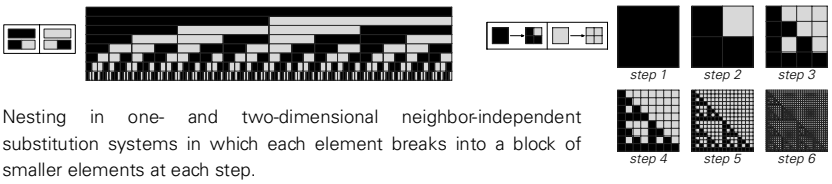
from step 1 from step 1000 from step 5000

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.

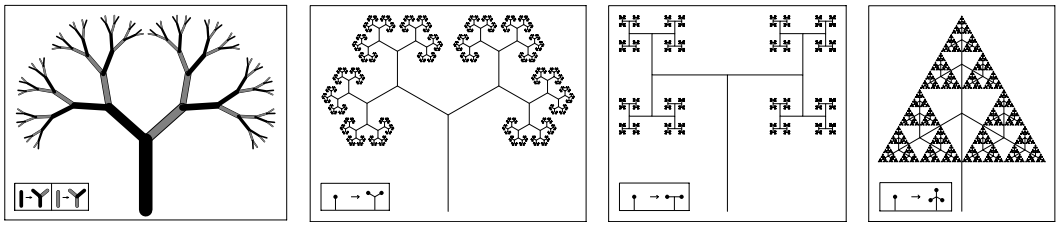
Beyond uniformity and repetition, the one further type of simple behavior that we have often encountered in this book is nesting. And as with uniformity and repetition, there are several quite different ways that nesting seems to arise.

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.



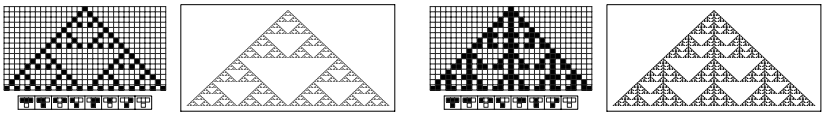
Nested patterns generated by simple branching processes. (Compare page 406.)

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.

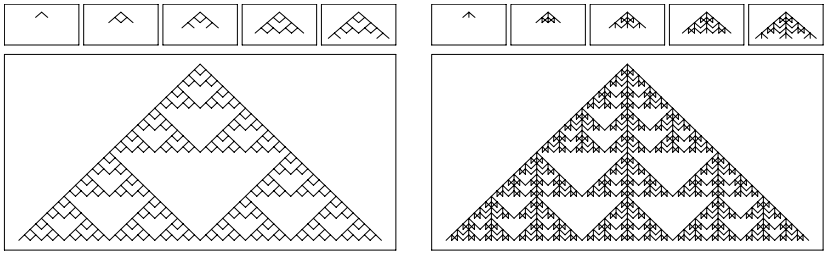
But in general nesting need not just arise from larger elements being broken down into smaller ones: for as we have discovered in this book it can also arise when larger elements are built up from smaller ones—and indeed I suspect that this is its more common origin in nature.

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.

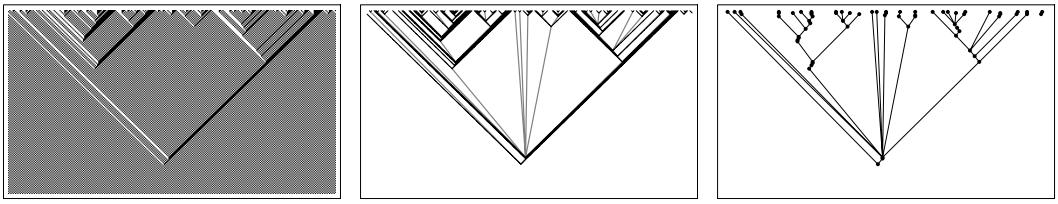
It turns out that the very same patterns can also be produced—as the pictures below illustrate—by processes in which new branches form at regular intervals, and annihilate when any pair of them collide.



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.

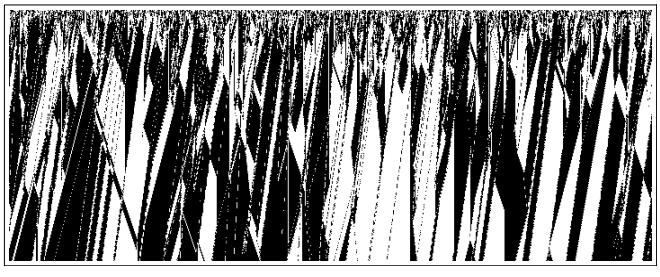


The generation of a nested pattern by rule 184 starting from random initial conditions. The pattern consists of a collection of stripes, highlighted in the second picture, which form the tree-like structure shown in the third picture. The initial condition used has exactly equal numbers of black and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

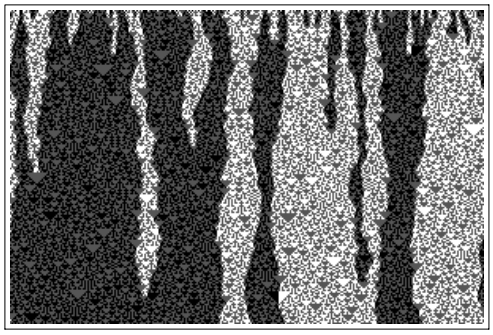


A highly compressed representation of the evolution of rule 110 from random initial conditions in which only the first cell in every 14 × 7 block is sampled.

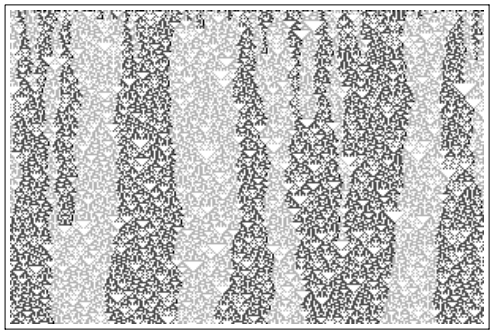
samples just the first cell in every 14 × 7 block of cells, making each domain of repetitive behavior stand out as having a uniform color.

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.