EXCERPTED FROM

STEPHEN WOLFRAM
A NEW KIND OF SCIENCE

SECTION 6.6

Special Initial Conditions

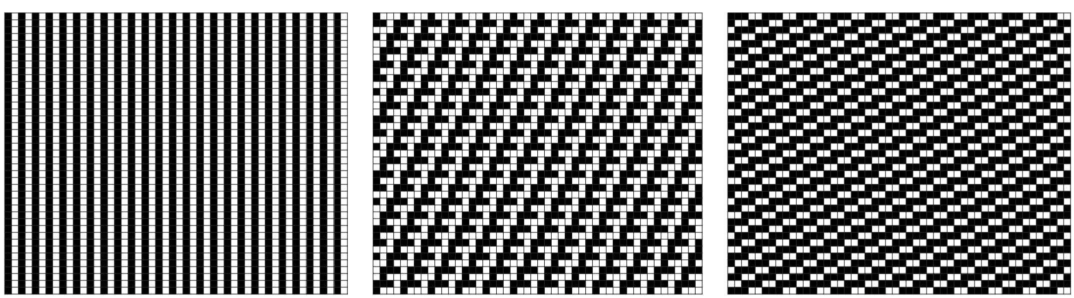
A couple of sections ago we saw that all class 3 systems have the property that the detailed patterns they produce are highly sensitive to detailed changes in initial conditions. But despite this sensitivity at the level of details, the point is that any system like rule 22 or rule 30 yields patterns whose overall properties depend very little on the form of the initial conditions that are given.

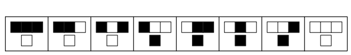
By intrinsically generating randomness such systems in a sense have a certain fundamental stability: for whatever is done to their initial conditions, they still give the same overall random behavior, with the same large-scale properties. And as we shall see in the next few chapters, there are in fact many systems in nature whose apparent stability is ultimately a consequence of just this kind of phenomenon.

Special Initial Conditions

We have seen that cellular automata such as rule 30 generate seemingly random behavior when they are started both from random initial conditions and from simple ones. So one may wonder whether there are in fact any initial conditions that make rule 30 behave in a simple way.

As a rather trivial example, one certainly knows that if its initial state is uniformly white, then rule 30 will just yield uniform white forever. But as the pictures below demonstrate, it is also possible to find less trivial initial conditions that still make rule 30 behave in a simple way.





Examples of special initial conditions that make the rule 30 cellular automaton yield simple repetitive behavior. Small patches with the same structures as shown here can be seen embedded in typical random patterns produced by rule 30. At left is a representation of rule 30. Finding initial conditions that make cellular automata yield behavior with certain repetition periods is closely related to the problem of satisfying constraints discussed on page 210.

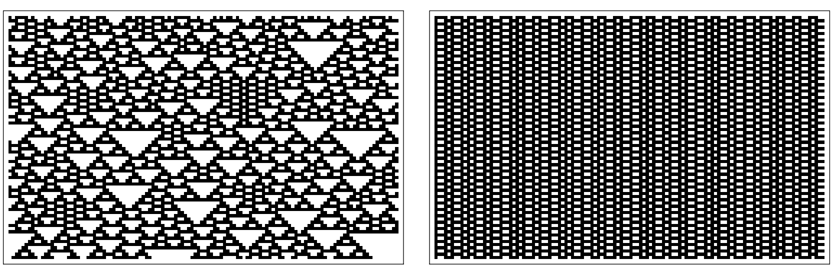
In fact, it turns out that in any cellular automaton it is inevitable that initial conditions which consist just of a fixed block of cells repeated forever will lead to simple repetitive behavior.

For what happens is that each block in effect independently acts like a system of limited size. The right-hand neighbor of the rightmost cell in any particular block is the leftmost cell in the next block, but since all the blocks are identical, this cell always has the same color as the leftmost cell in the block itself. And as a result, the block evolves just like one of the systems of limited size that we discussed on page 255. So this means that given a block that is n cells wide, the repetition period that is obtained must be at most 2^n steps.

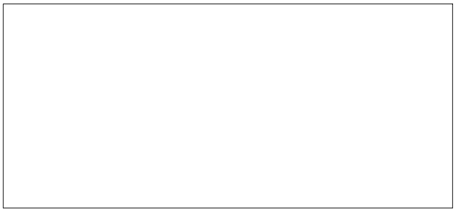
But if one wants a short repetition period, then there is a question of whether there is a block of any size which can produce it. The pictures on the next page show the blocks that are needed to get repetition periods of up to ten steps in rule 30. It turns out that no block of any size gives a period of exactly two steps, but blocks can be found for all larger periods at least up to 15 steps.

But what about initial conditions that do not just consist of a single block repeated forever? It turns out that for rule 30, no other kind of initial conditions can ever yield repetitive behavior.

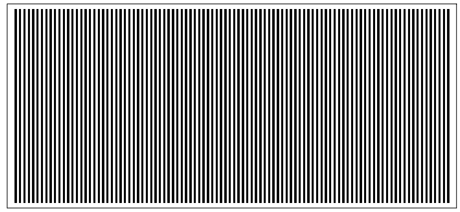
But for many rules—including a fair number of class 3 ones—the situation is different. And as one example the picture on the right below shows an initial condition for rule 126 that involves two different blocks but which nevertheless yields repetitive behavior.



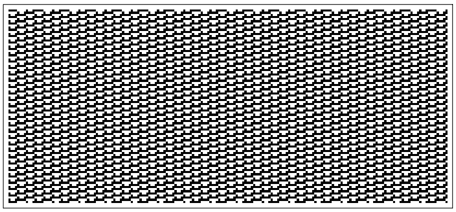
Rule 126 with a typical random initial condition, and with an initial condition that consists of a random sequence of the blocks ■■□□ and ■■■□. Rule 126 in general shows class 3 behavior, as on the left. But with the special initial condition on the right it acts like a simple class 2 rule. Note the patches of class 2 behavior even in the picture on the left.



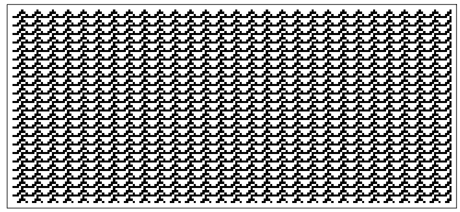
period 1, block size 1



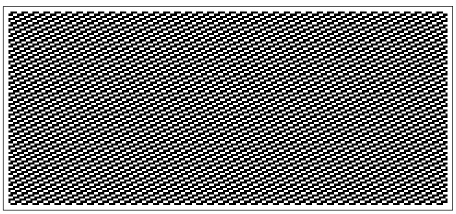
period 1, block size 2



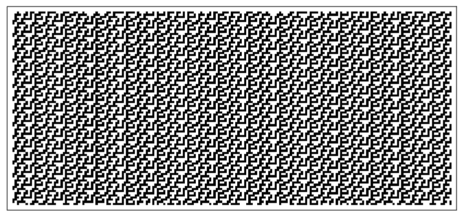
period 3, block size 12



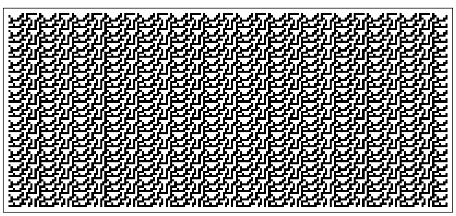
period 4, block size 7



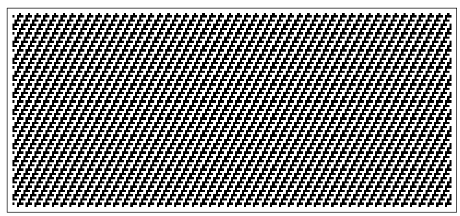
period 5, block size 5



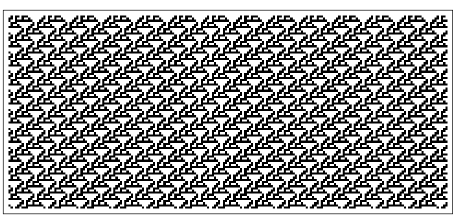
period 6, block size 84



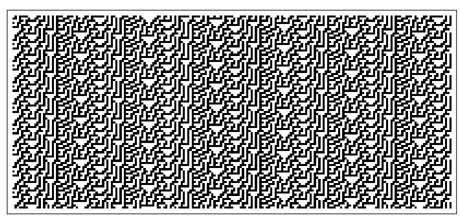
period 7, block size 15



period 8, block size 4



period 9, block size 15



period 10, block size 155

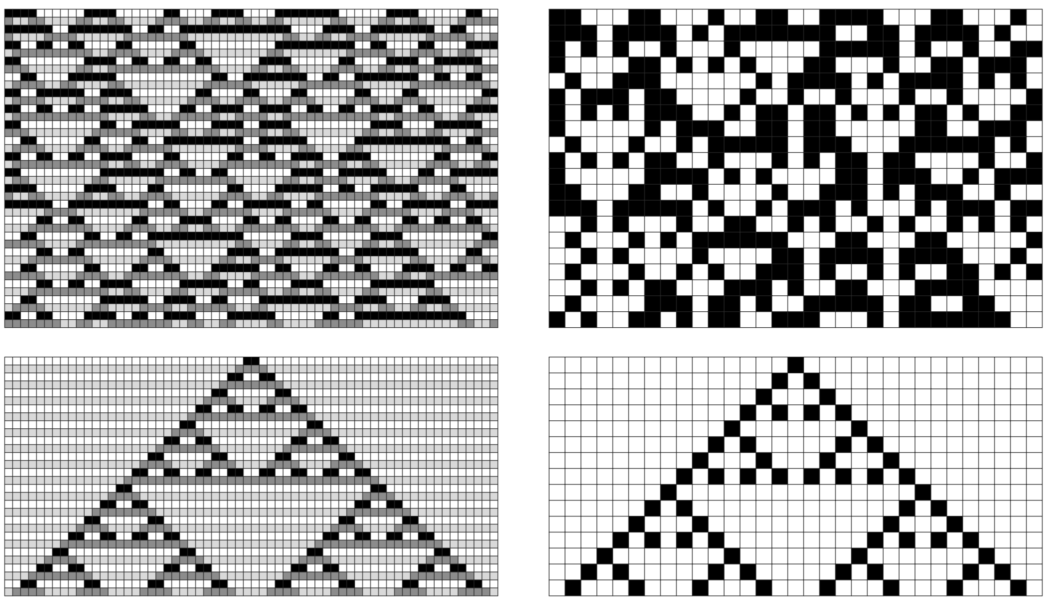
All patterns that repeat in 10 or less steps under evolution according to rule 30. In each case the initial conditions consist of a fixed block of cells that is repeated over and over again. Note that there are no initial conditions that yield a repetition period of exactly 2 steps. To get period 11, a block that contains 275 cells is required.

In a sense what is happening here is that even though rule 126 usually shows class 3 behavior, it is possible to find special initial conditions that make it behave like a simple class 2 rule.

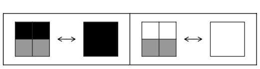
And in fact it turns out to be quite common for there to exist special initial conditions for one cellular automaton that make it behave just like some other cellular automaton.

Rule 126 will for example behave just like rule 90 if one starts it from special initial conditions that contain only blocks consisting of pairs of black and white cells.

The pictures below show how this works: on alternate steps the arrangement of blocks in rule 126 corresponds exactly to the arrangement of individual cells in rule 90. And among other things this explains why it is that with simple initial conditions rule 126 produces exactly the same kind of nested pattern as rule 90.



rule 126 rule 90



Two examples of the fact that with special initial conditions rule 126 behaves exactly like rule 90. The initial conditions that are used consist of blocks of cells where each block contains either two black cells or two white cells. If one looks only on every other step, then the blocks behave exactly like individual cells in rule 90. This correspondence is the basic reason that rule 126 produces the same kind of nested patterns as rule 90 when it is started from simple initial conditions.

The point is that these initial conditions in effect contain only blocks for which rule 126 behaves like rule 90. And as a result, the overall patterns produced by rule 126 in this case are inevitably exactly like those produced by rule 90.

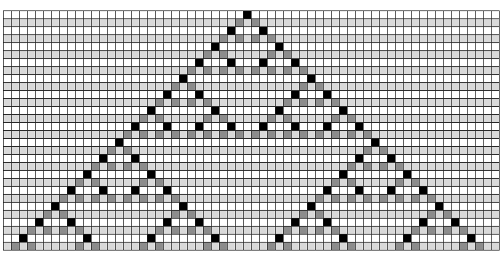
So what about other cellular automata that can yield similar patterns? In every example in this book where nested patterns like those from rule 90 are obtained it turns out that the underlying rules that are responsible can be set up to behave exactly like rule 90. Sometimes this will happen, say, for any initial condition that has black cells only in a limited region. But in other cases—like the example of rule 22 on page 263—rule 90 behavior is obtained only with rather specific initial conditions.

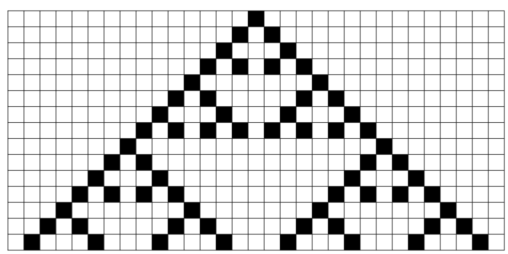
So what about rule 90 itself? Why does it yield nested patterns?

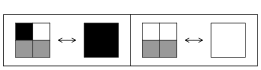
The basic reason can be thought of as being that just as other rules can emulate rule 90 when their initial conditions contain only certain blocks, so also rule 90 is able to emulate itself in this way.

The picture below shows how this works. The idea is to consider the initial conditions not as a sequence of individual cells, but rather as a sequence of blocks each containing two adjacent cells. And with an appropriate form for these blocks what one finds is that the configuration of blocks evolves exactly according to rule 90.

The fact that both individual cells and whole blocks of cells evolve according to the same rule then means that whatever pattern is



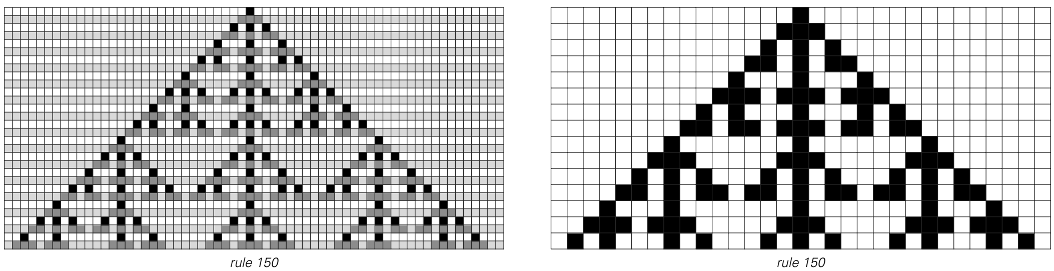


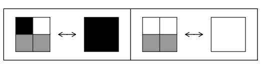


A demonstration of the fact that in rule 90 blocks of cells can behave just like individual cells. One consequence of this is that the patterns produced by rule 90 have a nested or self-similar form.

produced must have exactly the same structure whether it is looked at in terms of individual cells or in terms of blocks of cells. And this can be achieved in only two ways: either the pattern must be essentially uniform, or it must have a nested structure—just like we see in rule 90.

So what happens with other rules? It turns out that the property of self-emulation is rather rare among cellular automaton rules. But one other example is rule 150—as illustrated in the picture below.

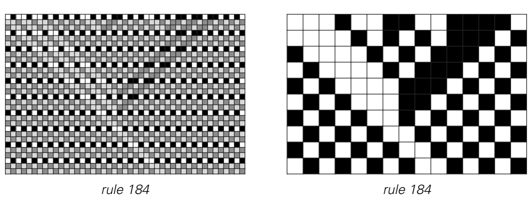


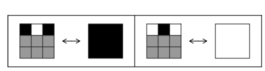


Another example of a rule in which blocks of cells can behave just like individual cells. Rule 90 and rule 150 are also essentially the only fundamentally different elementary cellular automaton rules that have the property of being additive (see page 264).

So what else is there in common between rule 90 and rule 150? It turns out that they are both additive rules, implying that the patterns they produce can be superimposed in the way we discussed on page 264. And in fact one can show that any rule that is additive will be able to emulate itself and will thus yield nested patterns. But there are rather few additive rules, and indeed with two colors and nearest neighbors the only fundamentally different ones are precisely rules 90 and 150.

Ultimately, however, additive rules are not the only ones that can emulate themselves. An example of another kind is rule 184, in which blocks of three cells can act like a single cell, as shown below.

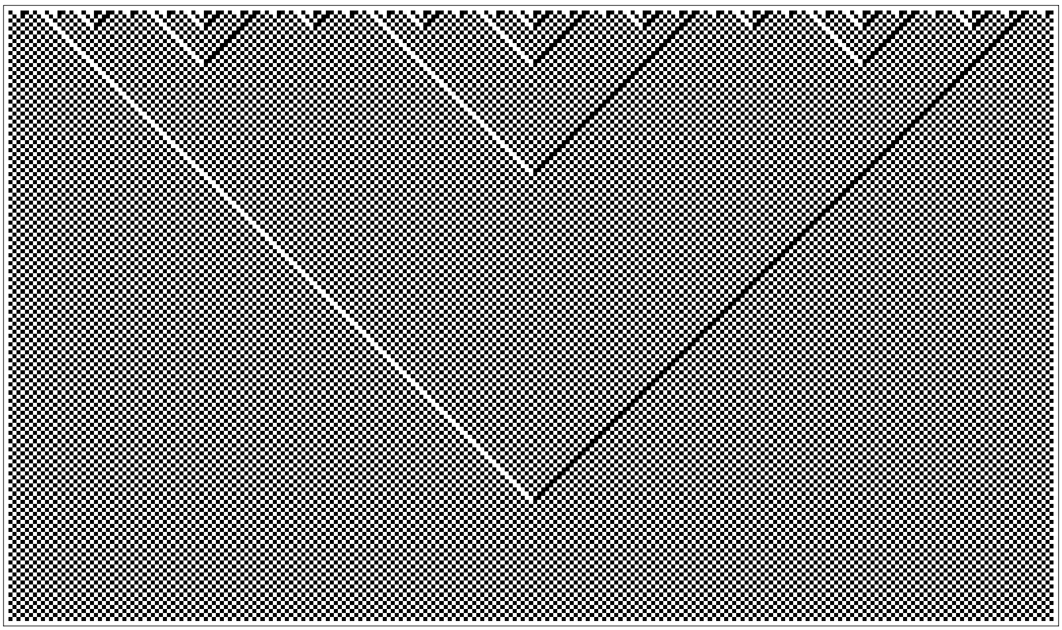


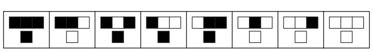


A rule that is not additive, but in which blocks of cells can again behave just like individual cells.

With simple initial conditions of the type we have used so far this rule will always produce essentially trivial behavior. But one way to see the properties of the rule is to use nested initial conditions, obtained for example from substitution systems of the kind we discussed on page 82.

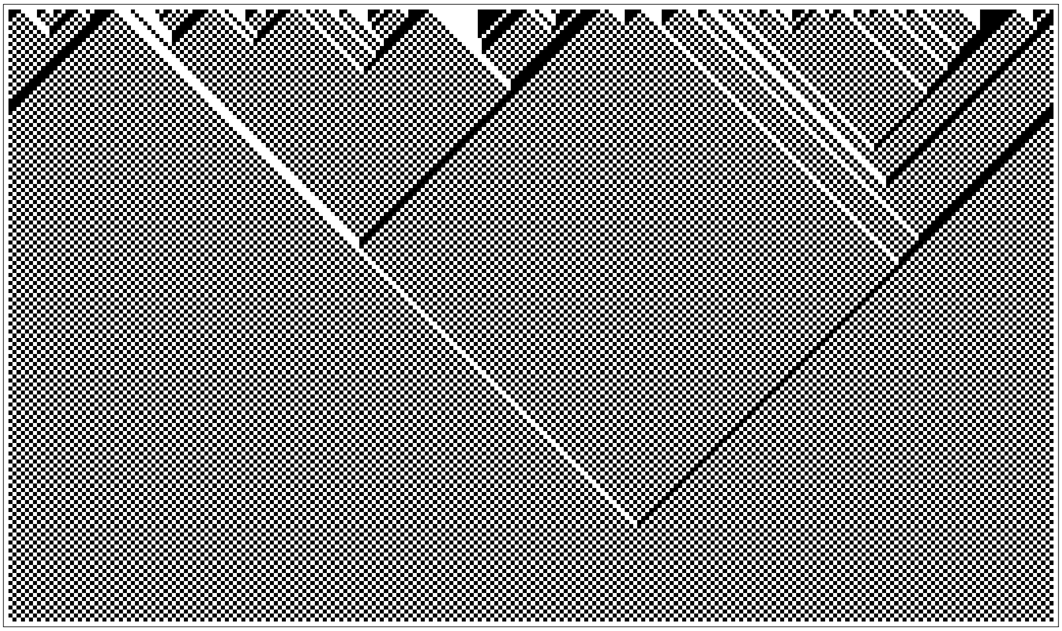
With most rules, including 90 and 150, such nested initial conditions typically yield results that are ultimately indistinguishable from those obtained with typical random initial conditions. But for rule 184, an appropriate choice of nested initial conditions yields the highly regular pattern shown below.





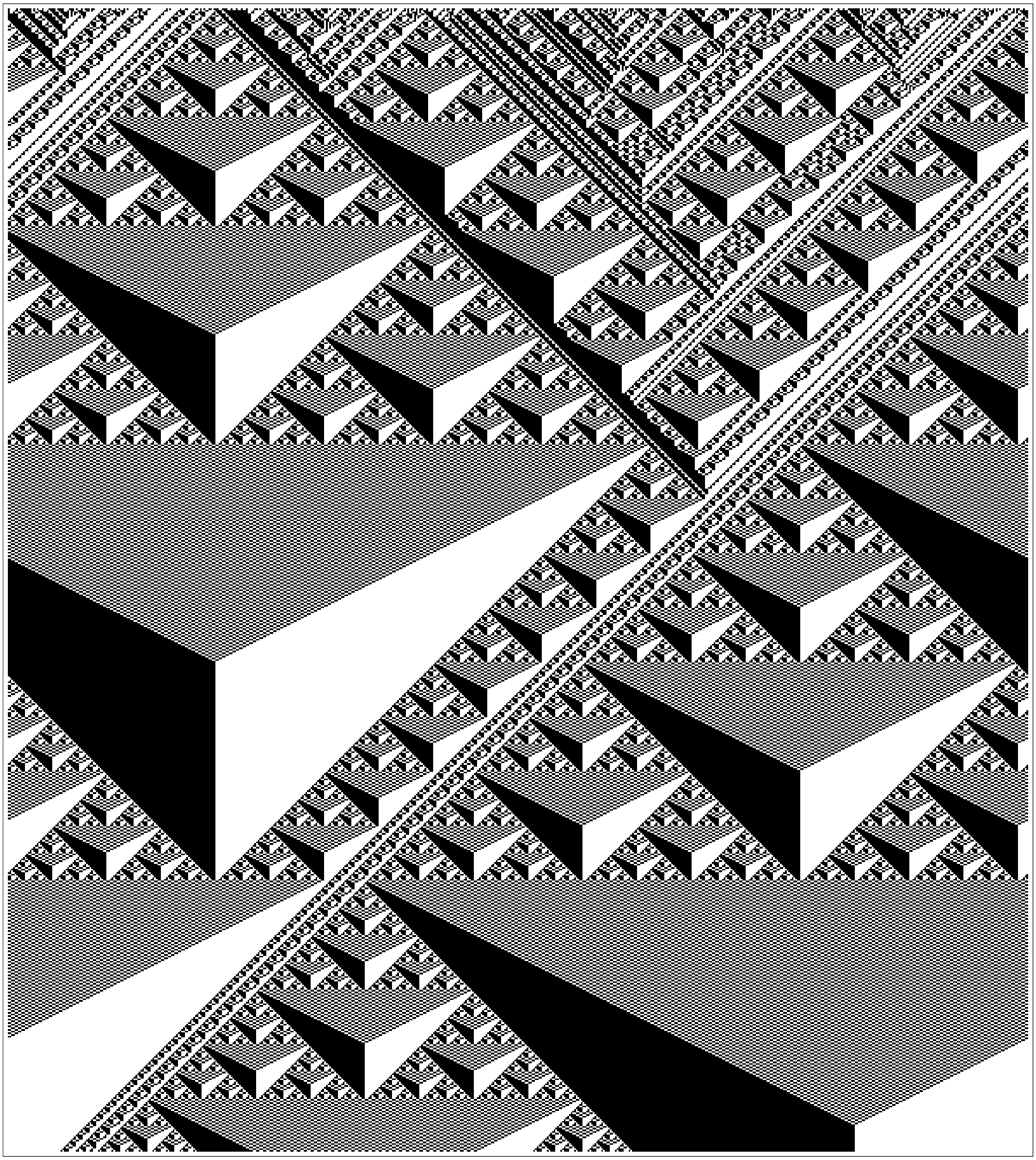
The pattern produced by rule 184 (shown at left) evolving from a nested initial condition. The particular initial condition shown can be obtained by applying the substitution system ■ → ■□■, □ → □□■, starting from a single black element ■ (see page 83). With this initial condition, rule 184 exhibits an equal number of black and white stripes, which annihilate in pairs so as to yield a regular nested pattern.

The nested structure seen in this pattern can then be viewed as a consequence of the fact that rule 184 is able to emulate itself. And the picture below shows that rule 184—unlike any of the additive rules—still produces recognizably nested patterns even when the initial conditions that are used are random.



Rule 184 evolving from a random initial condition. Nested structure similar to what we saw in the previous picture is still visible. The presence of such structure is most obvious when there are equal numbers of black and white cells in the initial conditions, but it does not rely on any regularity in the arrangement of these cells.

As we will see on page 338 the presence of such patterns is particularly clear when there are equal numbers of black and white cells in the initial conditions—but how these cells are arranged does not usually matter much at all. And in general it is possible to find quite a few cellular automata that yield nested patterns like rule 184 even from random initial conditions. The picture on the next page shows a particularly striking example in which explicit regions are formed that contain patterns with the same overall structure as rule 90.



Another example of a cellular automaton that produces a nested pattern even from random initial conditions. The particular rule shown involves next-nearest as well as nearest neighbors and has rule number 4067213884. As in rule 184, the nested behavior seen here is most obvious when the density of black and white cells in the initial conditions is equal.