EXCERPTED FROM

STEPHEN WOLFRAM
A NEW KIND OF SCIENCE

SECTION 5.7

Systems Based on Constraints

Systems Based on Constraints

In the course of this book we have looked at many different kinds of systems. But in one respect all these systems have ultimately been set up in the same basic way: they are all based on explicit rules that specify how the system evolves from step to step.

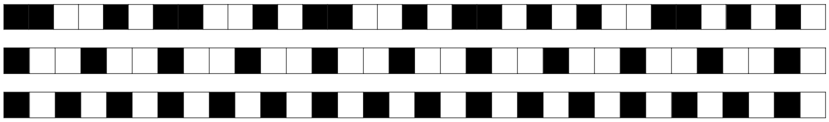
In traditional science, however, it is common to consider systems that are set up in a rather different way: instead of having explicit rules for evolution, the systems are just given constraints to satisfy.

As a simple example, consider a line of cells in which each cell is colored black or white, and in which the arrangement of colors is subject to the constraint that every cell should have exactly one black and one white neighbor. Knowing only this constraint gives no explicit procedure for working out the color of each cell. And in fact it may at first not be clear that there will be any arrangement of colors that can satisfy the constraint. But it turns out that there is—as shown below.

A system consisting of a line of black and white cells whose form is defined by the constraint that every cell should have exactly one black and one white neighbor. The pattern shown is the only possible one that satisfies this constraint. The idea of implicitly determining the behavior of a system by giving constraints that it must satisfy is common in traditional science and mathematics.

And having seen this picture, one might then imagine that there must be many other patterns that would also satisfy the constraint. After all, the constraint is local to neighboring cells, so one might suppose that parts of the pattern sufficiently far apart should always be independent. But in fact this is not true, and instead the system works a bit like a puzzle in which there is only one way to fit in each piece. And in the end it is only the perfectly repetitive pattern shown above that can satisfy the required constraint at every cell.

Other constraints, however, can allow more freedom. Thus, for example, with the constraint that every cell must have at least one neighbor whose color is different from its own, any of the patterns in the picture at the top of the facing page are allowed, as indeed is any pattern that involves no more than two successive cells of the same color.



A system consisting of a line of black and white cells whose form is defined by the constraint that every cell should have at least one neighbor whose color is different from its own. There are many possible arrangements of colors that satisfy this constraint. Some, like the first arrangement above, look quite random. But others, like the second two arrangements above, are simple and repetitive. It turns out that in a one-dimensional system no set of local constraints can force arrangements of more complicated types.

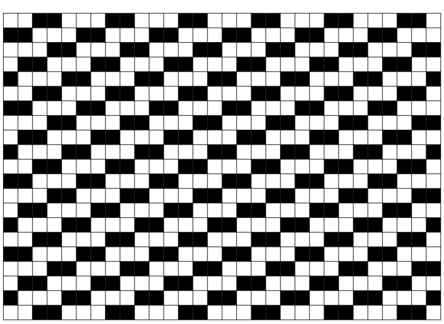
But while the first arrangement of colors shown above looks somewhat random, the last two are simple and purely repetitive.

So what about other choices of constraints? We have seen in this book many examples of systems where simple sets of rules give rise to highly complex behavior. But what about systems based on constraints? Are there simple sets of constraints that can force complex patterns?

It turns out that in one-dimensional systems there are not. For in one dimension it is possible to prove that any local set of constraints that can be satisfied at all can always be satisfied by some simple and purely repetitive arrangement of colors.

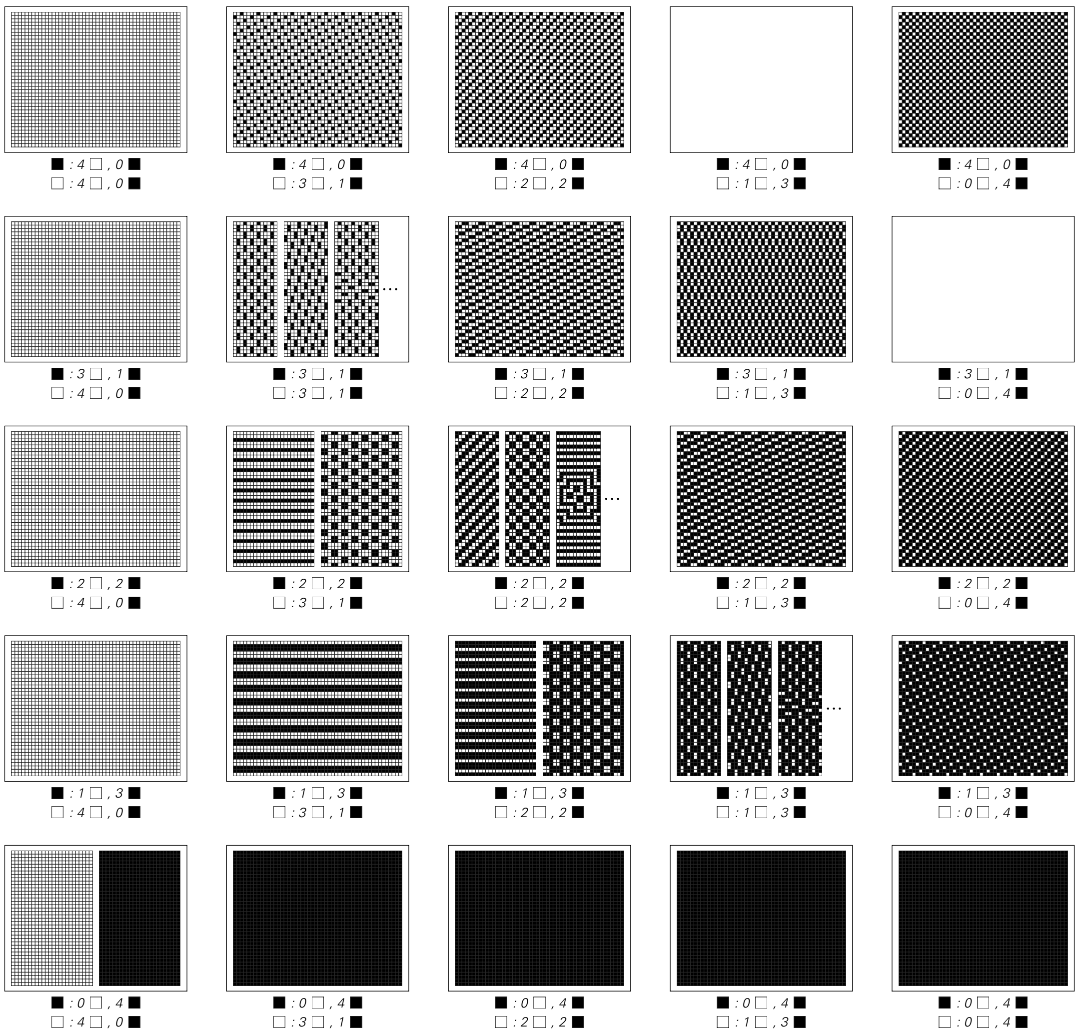
But what about two dimensions? The proof for one dimension breaks down in two dimensions, and so it becomes at least conceivable that a simple set of constraints could force a complex pattern to occur.

As a first example of a two-dimensional system, consider an array of black and white cells in which the constraint is imposed that every black cell should have exactly one black neighbor, and every white cell should have exactly two white neighbors.



A system consisting of a grid of black and white cells defined by the constraint that every black cell should have exactly one black neighbor among its four neighbors, and every white cell should have exactly two white neighbors. The infinite repetitive pattern shown here, together with its rotations and reflections, is the only one that satisfies this constraint. (The picture is assumed to wrap around at each edge.) The pattern can be viewed as a tessellation of 5 × 5 blocks of cells.

As in one dimension, knowing the constraint does not immediately provide a procedure for finding a pattern which satisfies it. But a little experimentation reveals that the simple repetitive pattern above satisfies the constraint, and in fact it is the only pattern to do so.

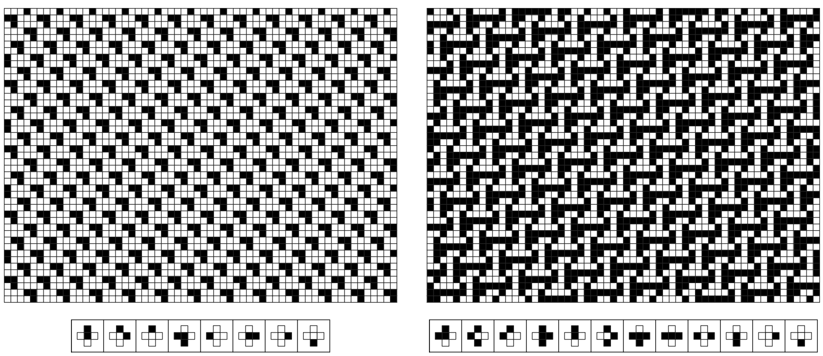


Patterns satisfying constraints which specify that every black cell and every white cell must have a certain fixed number of black and white neighbors. The blank rectangles in the upper right indicate constraints that cannot be satisfied by any pattern whatsoever. Most of the constraints are satisfied by a single pattern, together with its rotations and reflections. In some cases, two distinct patterns are possible, and in a few cases, an infinite set of patterns are possible. In all cases where the constraints can be satisfied at all, a simple repetitive pattern nevertheless suffices.

What about other constraints? The pictures on the facing page show schematically what happens with constraints that require each cell to have various numbers of black and white neighbors.

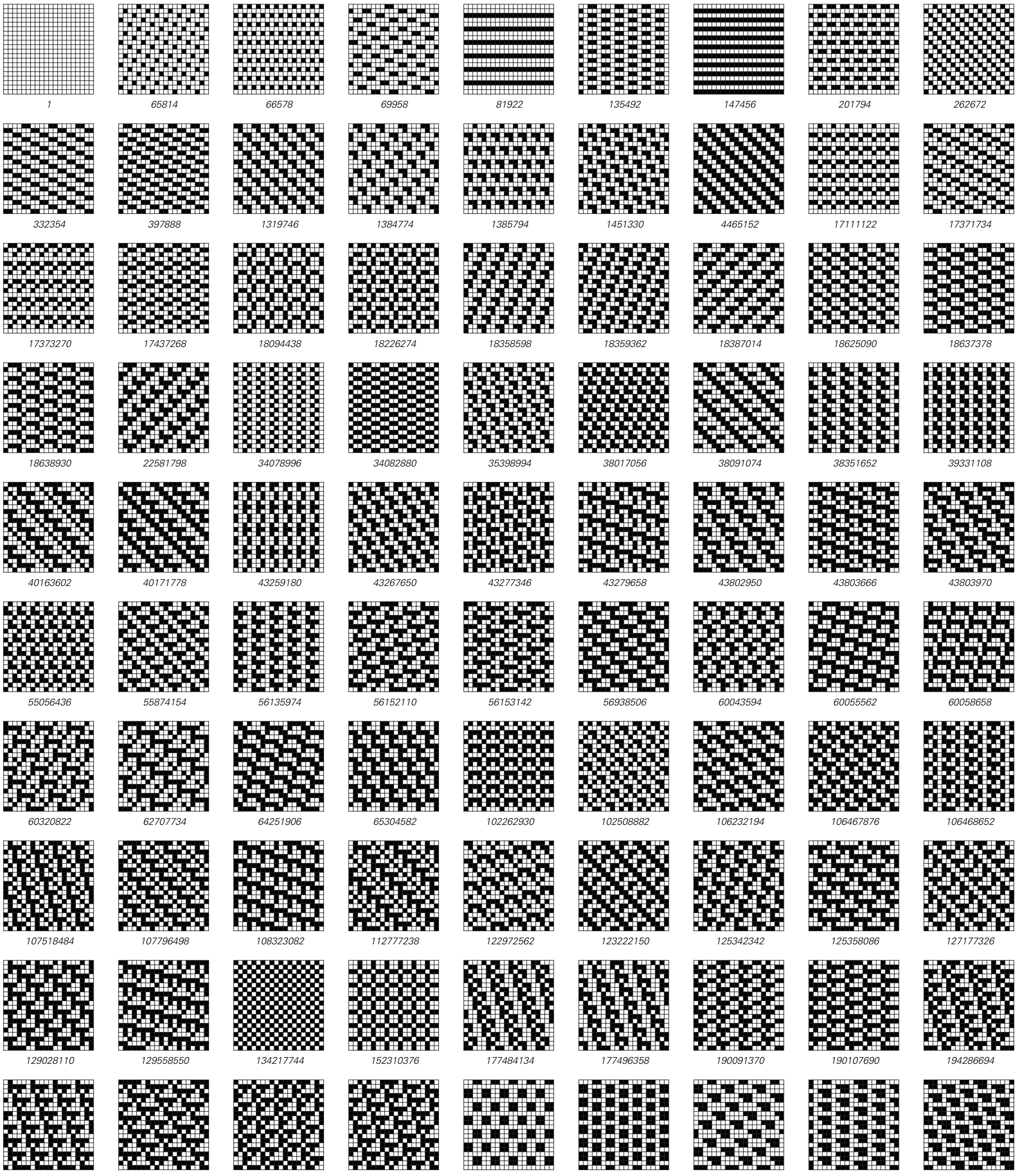
Several kinds of results are seen. In the two cases shown as blank rectangles on the upper right, there are no patterns at all that satisfy the constraints. But in every other case the constraints can be satisfied, though typically by just one or sometimes two simple infinite repetitive patterns. In the three cases shown in the center a whole range of mixtures of different repetitive patterns are possible. But ultimately, in every case where some pattern can work, a simple repetitive pattern is all that is needed.

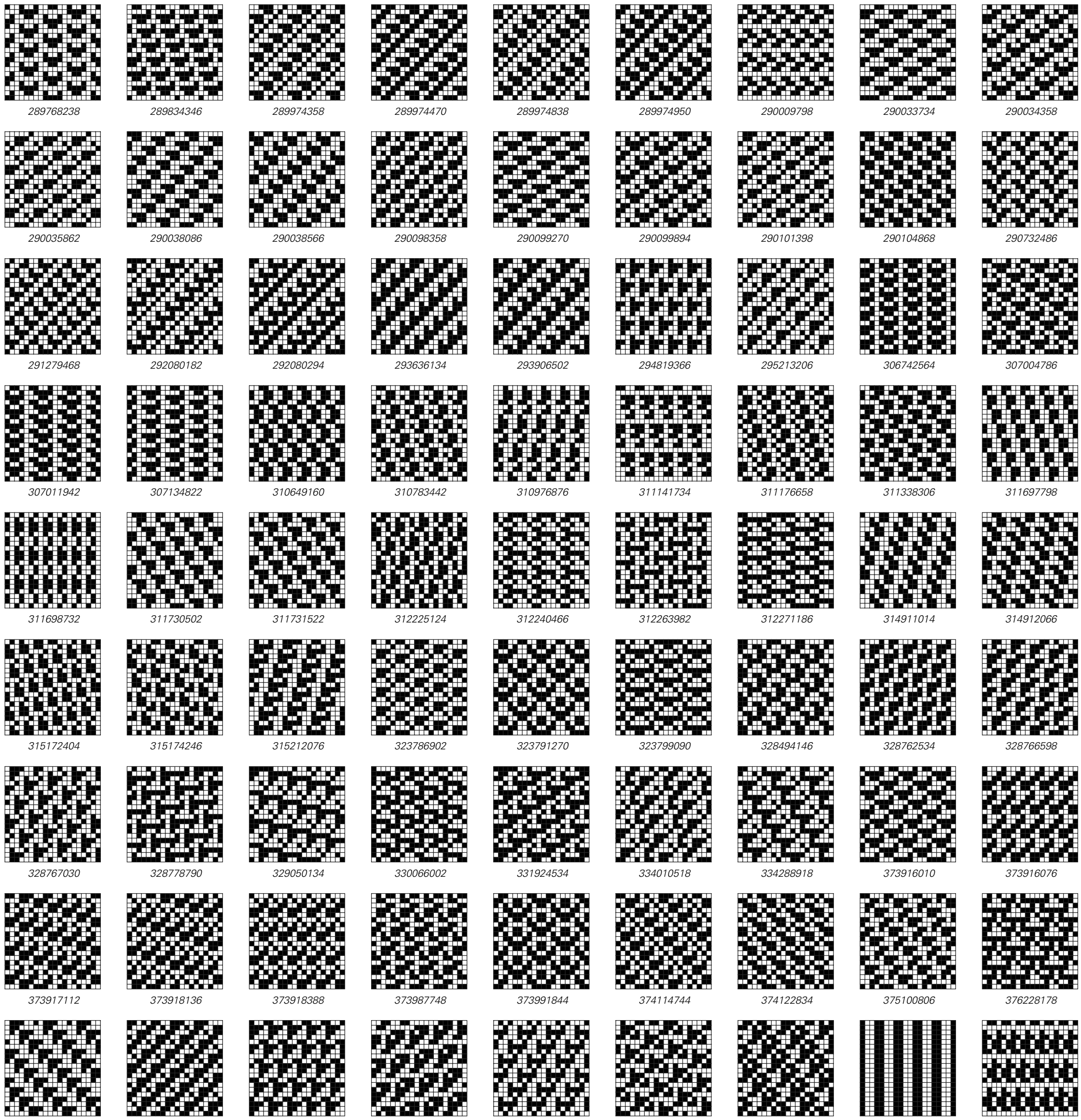
So what about more complicated constraints? The pictures below show examples based on constraints that require the local arrangement of colors around every cell to match a fixed set of possible templates.



Systems specified by the constraint that the local arrangement of colors around every cell must match the fixed set of possible templates shown. Note that these templates apply to every cell, with templates of neighboring cells overlapping. Pattern (a) can be viewed as formed from a tessellation of 5 × 10 blocks of cells; pattern (b) from a tessellation of 24 × 24 blocks. With the numbering scheme for constraints used on the next two pages the cases shown here correspond to 1384774 and 328778790.

There are a total of 4,294,967,296 possible sets of such templates. And of these, 766,979,044 lead to constraints that cannot be satisfied by any pattern. But among the 3,527,988,252 that remain, it turns out that every single one can be satisfied by a simple repetitive pattern. In fact the number of different repetitive patterns that are ever needed is quite small: if a particular constraint can be satisfied by any pattern, then one of the set of 171 repetitive patterns on the next two pages is always sufficient.


1 65814 66578 69958 81922 135492 147456 201794 262672
332354 397888 1319746 1384774 1385794 1451330 4465152 17111122 17371734
17373270 17437268 18094438 18226274 18358598 18359362 18387014 18625090 18637378
18638930 22581798 34078996 34082880 35398994 38017056 38091074 38351652 39331108
40163602 40171778 43259180 43267650 43277346 43279658 43802950 43803666 43803970
55056436 55874154 56135974 56152110 56153142 56938506 60043594 60055562 60058658
60320822 62707734 64251906 65304582 102262930 102508882 106232194 106467876 106468652
107518484 107796498 108323082 112777238 122972562 123222150 125342342 125358086 127177326
129028110 129558550 134217744 152310376 177484134 177496358 190091370 190107690 194286694
194303014 257478694 261132398 261148718 272703878 272770436 272998726 273064262 273065282

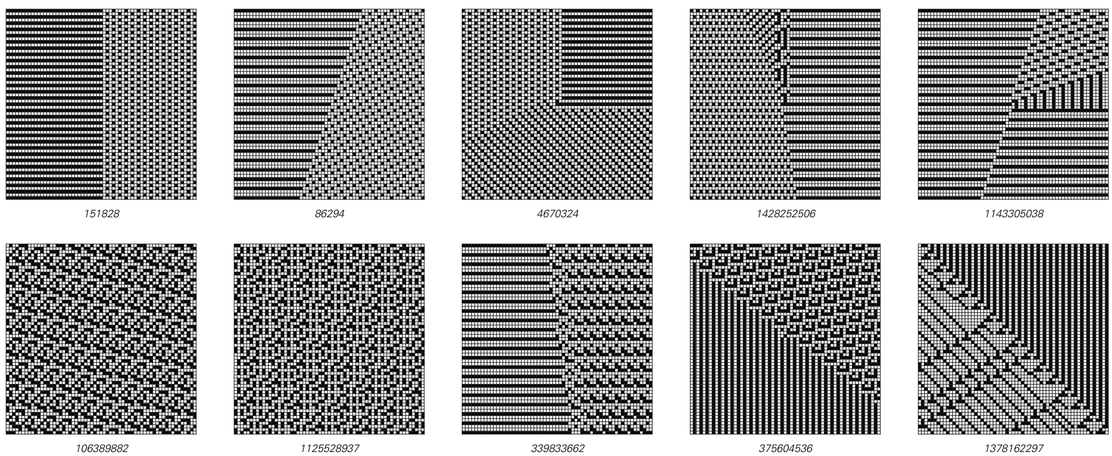


The complete collection of all 171 patterns needed to satisfy constraints of the type shown on the previous page. If none of these 171 patterns satisfy a particular constraint, then it follows that no pattern at all will satisfy the constraint. The patterns are labelled by numbers which specify the minimal constraint which requires the given pattern. Patterns differing by overall reflection, rotation or interchange of black and white are not shown.

So how can one force more complex patterns to occur?

The basic answer is that one must extend at least slightly the kinds of constraints that one considers. And one way to do this is to require not only that the colors around each cell match a set of templates, but also that a particular template from this set must appear at least somewhere in the array of cells.

The pictures below show a few examples of patterns determined by constraints of this kind. A typical feature is that the patterns are divided into several separate regions, often emanating from some kind of center. But at least in all the examples below, the patterns that occur in each individual region are still simple and repetitive.



Examples of patterns produced by systems in which not only must the arrangement of colors in each neighborhood match one of a fixed set of templates, but also a certain template from this set must occur at least once in the pattern. The constraints are numbered as before, and in each picture the template that must occur is shown at the center. Constraint 1125528937 leads to a pattern that repeats in 98 × 98 blocks. The last pattern shown is also repetitive, repeating every 56 cells on the diagonal.

So how can one find constraints that force more complex patterns? To do so has been fairly difficult, and in fact has taken almost as much computational effort as any other single result in this book.

The basic problem is that given a constraint it can be extremely difficult to find out what pattern—if any—will satisfy the constraint.

In a system like a cellular automaton that is based on explicit rules, it is always straightforward to take the rule and apply it to see what pattern is produced. But in a system that is based on constraints, there is no such direct procedure, and instead one must in effect always go outside of the system to work out what patterns can occur.

The most straightforward approach might just be to enumerate every single possible pattern and then see which, if any, of them satisfy a particular constraint. But in systems containing more than just a few cells, the total number of possible patterns is absolutely astronomical, and so enumerating them becomes completely impractical.

A more practical alternative is to build up patterns iteratively, starting with a small region, and then adding new cells in essentially all possible ways, at each stage backtracking if the constraint for the system does not end up being satisfied.

The pictures on the next page show a few sequences of patterns produced by this method. In some cases, there emerge quite quickly simple repetitive patterns that satisfy the constraint. But in other cases, a huge number of possibilities have to be examined in order to find any suitable pattern.

And what if there is no pattern at all that can satisfy a particular constraint? One might think that to demonstrate this would effectively require examining every conceivable pattern on the infinite grid of cells. But in fact, if one can show that there is no pattern that satisfies the constraint in a limited region, then this proves that no pattern can satisfy the constraint on the whole grid. And indeed for many constraints, there are already quite small regions for which it is possible to establish that no pattern can be found.

But occasionally, as in the third picture on the next page, one runs into constraints that can be satisfied for regions containing thousands of cells, but not for the whole grid. And to analyze such cases inevitably requires examining huge numbers of possible patterns.

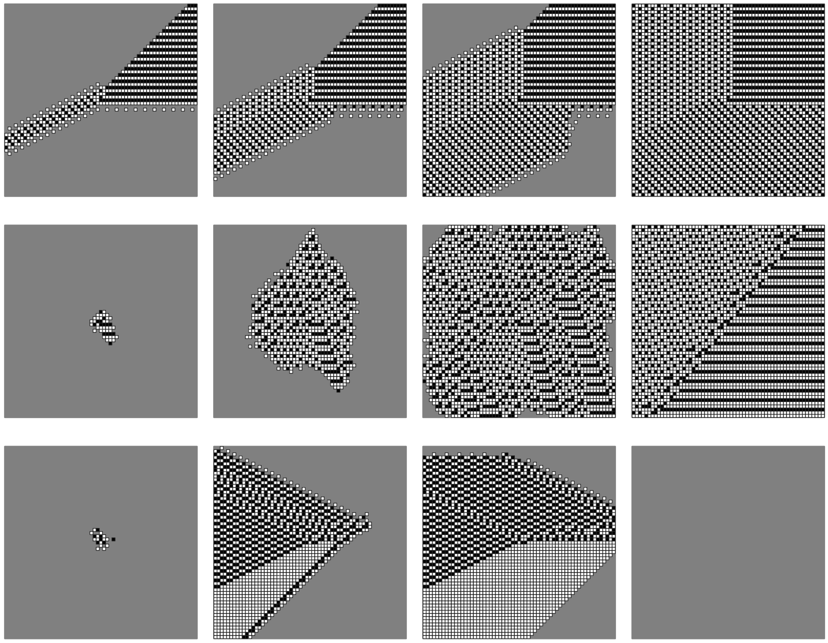
But with an appropriate collection of tricks, it is in the end feasible to take almost any system of the type discussed here, and determine what pattern, if any, satisfies its constraint.

So what kinds of patterns can be needed? In the vast majority of cases, simple repetitive patterns, or mixtures of such patterns, are the only ones that are needed.

(a)

(b)

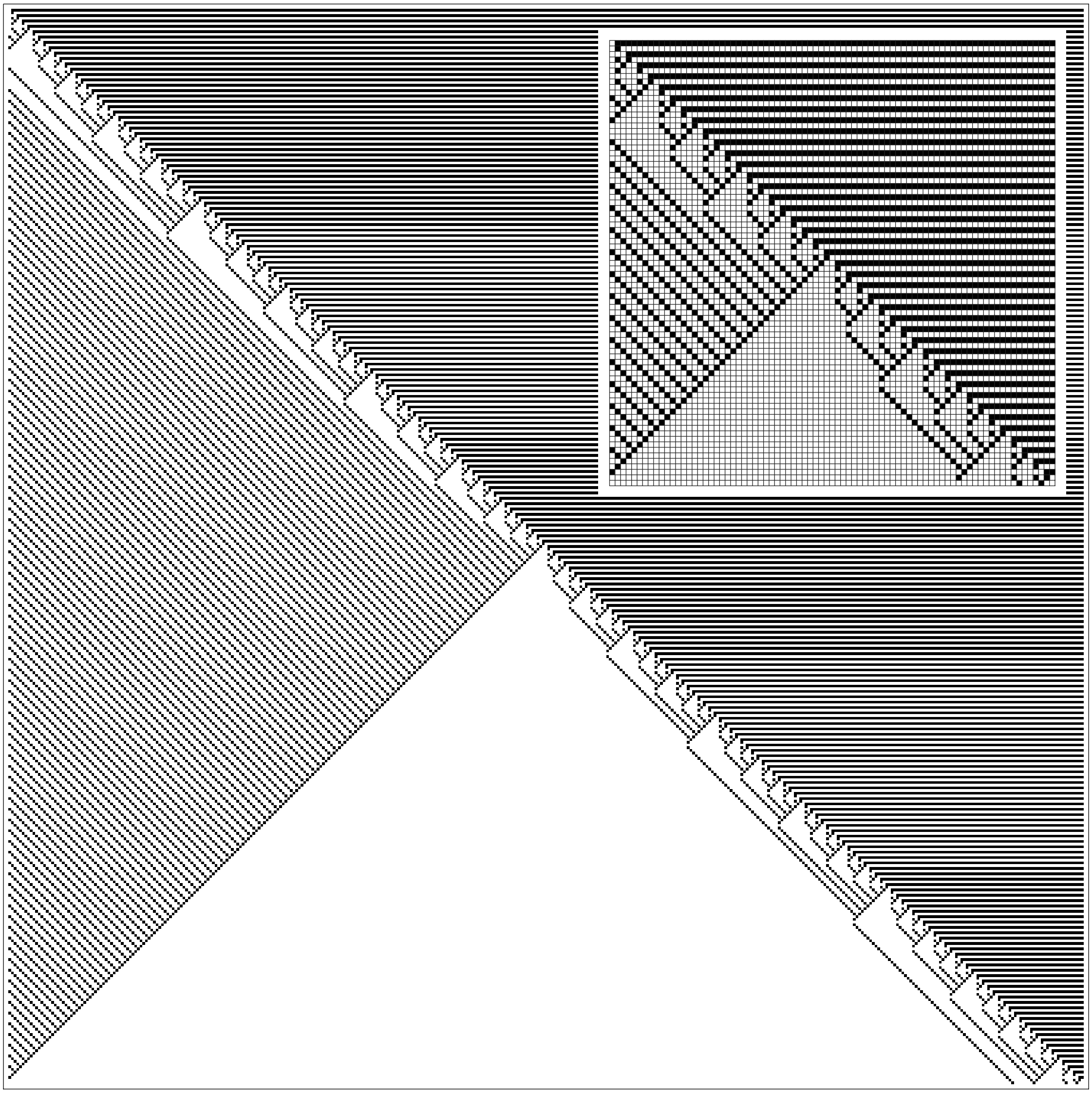
(c)

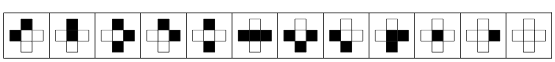


Stages in finding patterns that satisfy constraints (a) 4670324, (b) 373384574, and (c) 387520105. Gray is used to indicate cells whose colors have not yet been determined. The first stage shown in each case corresponds to cells whose colors can be deduced immediately from the presence of a particular template at the center. In case (a) choices for additional cells can be made straightforwardly, and an infinite regular pattern can be built up without any backtracking. In case (b), many choices for additional cells have to be tried, with much backtracking, and in the end the automatic procedure fails to find a repetitive pattern. Nevertheless, as the last stage demonstrates, a repetitive pattern does in fact exist. In case (c), the automatic procedure finds a fairly large and almost regular pattern that satisfies the constraints, but in this case it turns out that no infinite pattern exists.

But if one systematically examines possible constraints in the order shown on pages 214 and 215, then it turns out that after examining more than 18 million of them, one finally discovers the system shown on the facing page. And in this system, unlike all others before it, no repetitive pattern is possible; the only pattern that satisfies the constraint is the non-repetitive nested pattern shown in the picture.

After testing millions of constraints, and tens of billions of candidate patterns, therefore, it is finally possible to establish that a system based on simple constraints of the type discussed here can be forced to exhibit behavior more complex than pure repetition.



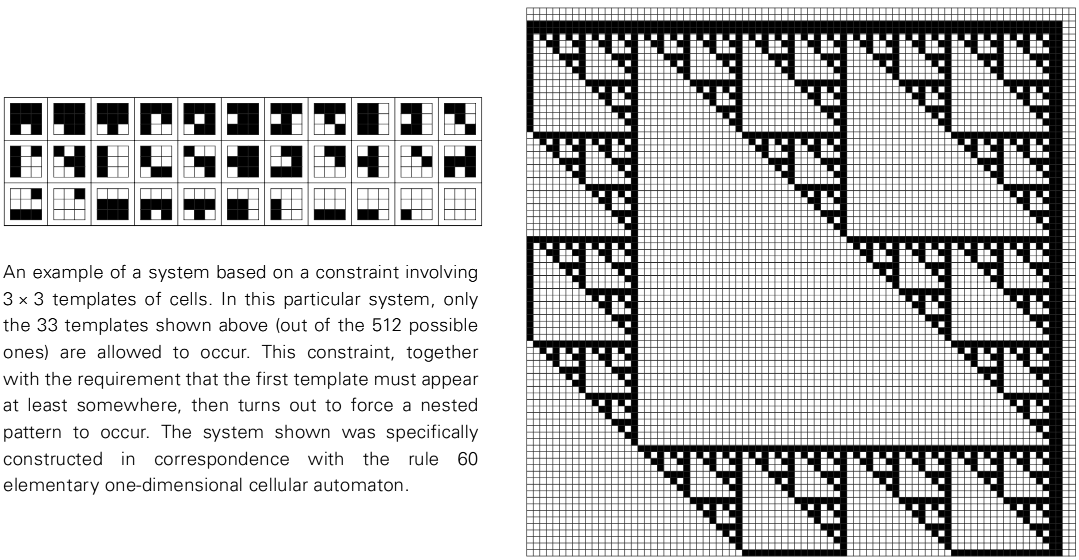


The simplest system based on constraints that is forced to exhibit a non-repetitive pattern. The constraint requires that the arrangement of colors around each cell must match one of the 12 templates shown, and that at least somewhere in the pattern a template containing a pair of stacked black cells must occur. In the numbering scheme used on preceding pages, the constraint is number 18762389. The pattern shown is unique, in that no variations of it, except for trivial translations, will satisfy the constraints. The nested structure on the diagonal essentially corresponds to a progression of base 2 digit sequences for positive and negative numbers.

What about still more complex behavior?

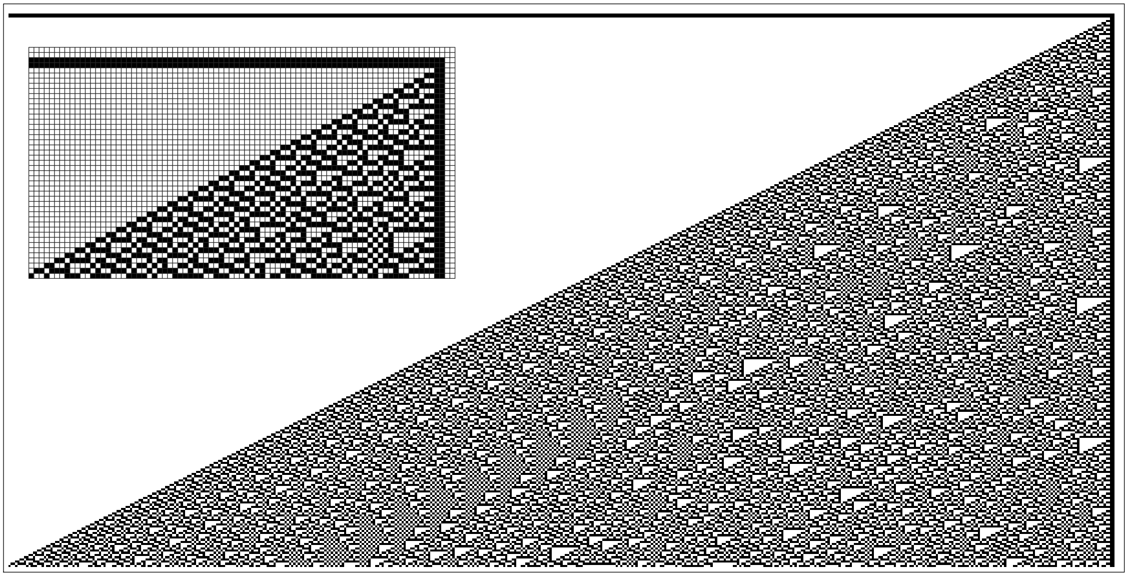
There are altogether 137,438,953,472 constraints of the type shown on page 216. And of the millions of these that I have tested, none have forced anything more complicated than the kind of nested behavior seen on the previous page. But if one extends again the type of constraints one considers, it turns out to become possible to construct examples that force more complex behavior.

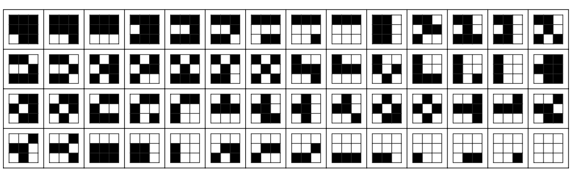
The idea is to set up templates that involve complete 3 × 3 blocks of cells, including diagonal neighbors. The picture below then shows an example of such a system, in which by allowing only a specific set of 33 templates, a nested pattern is forced to occur.



An example of a system based on a constraint involving 3 × 3 templates of cells. In this particular system, only the 33 templates shown above (out of the 512 possible ones) are allowed to occur. This constraint, together with the requirement that the first template must appear at least somewhere, then turns out to force a nested pattern to occur. The system shown was specifically constructed in correspondence with the rule 60 elementary one-dimensional cellular automaton.

What about more complex patterns? Searches have not succeeded in finding anything. But explicit construction, based on correspondence with one-dimensional cellular automata, leads to the example shown at the top of the facing page: a system with 56 allowed templates in which the only pattern satisfying the constraint is a complex and largely random one, derived from the rule 30 cellular automaton.





A system based on a constraint, in which a complex and largely random pattern is forced to occur. The constraint specifies that only the 56 3 × 3 templates shown at left can occur anywhere in the pattern, with the first template appearing at least once. The pattern required to satisfy this constraint corresponds to a shifted version of the one generated by the evolution of the rule 30 elementary one-dimensional cellular automaton.

So finally this shows that it is indeed possible to force complex behavior to occur in systems based on constraints. But from what we have seen in this section such behavior appears to be quite rare: unlike many of the simple rules that we have discussed in this book, it seems that almost all simple constraints lead only to fairly simple patterns.

Any phenomenon based on rules can always ultimately also be described in terms of constraints. But the results of this section indicate that these descriptions can have to be fairly complicated for complex behavior to occur. So the fact that traditional science and mathematics tends to concentrate on equations that operate like constraints provides yet another reason for their failure to identify the fundamental phenomenon of complexity that I discuss in this book.