EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

SECTION 11.4

A Universal Cellular Automaton

experience with computer languages, there is already an indication that the range of systems that are universal might be somewhat broader.

Indeed, *Mathematica* turns out to be a particularly good example, in which one can pick very different sets of operations to use, and yet still be able to implement exactly the same kinds of programs.

So what about cellular automata and other systems with simple rules? Is it possible for these kinds of systems to be universal?

At first, it seems quite implausible that they could be. For the intuition that one gets from practical computers and computer languages seems to suggest that to achieve universality there must be some fundamentally fairly sophisticated elements present.

But just as we found that the intuition which suggests that simple rules cannot lead to complex behavior is wrong, so also the intuition that simple rules cannot be universal also turns out to be wrong. And indeed, later in this chapter, I will show an example of a cellular automaton with an extremely simple underlying rule that can nevertheless in the end be seen to be universal.

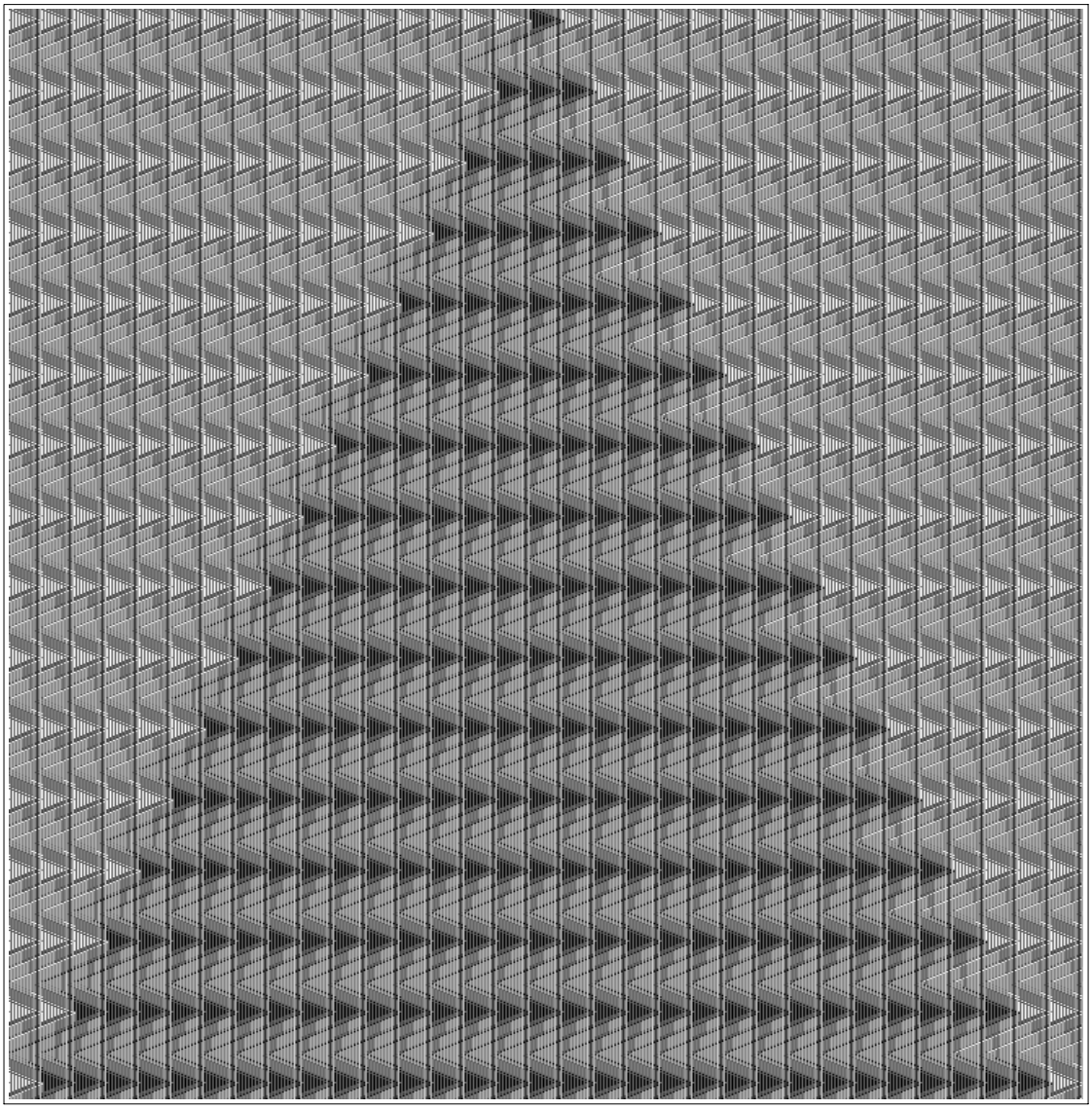
In the past it has tended to be assumed that universality is somehow a rare and special quality, usually possessed only by systems that are specifically constructed to have it. But one of the results of this chapter is that in fact universality is a much more widespread phenomenon. And in the next chapter I will argue that for example it also occurs in a wide range of important systems that we see in nature.

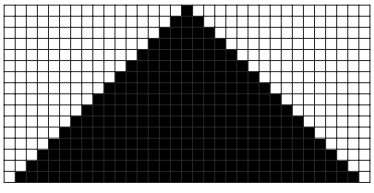
A Universal Cellular Automaton

As our first specific example of a system that exhibits universality, I discuss in this section a particular universal cellular automaton that has been set up to make its operation as easy to follow as possible.

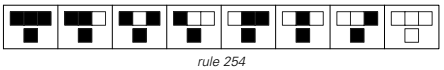
The rules for this cellular automaton itself are always the same. But the fact that it is universal means that if it is given appropriate initial conditions it can effectively be programmed to emulate for example any possible cellular automaton—with any set of rules.

The next three pages show three examples of this.

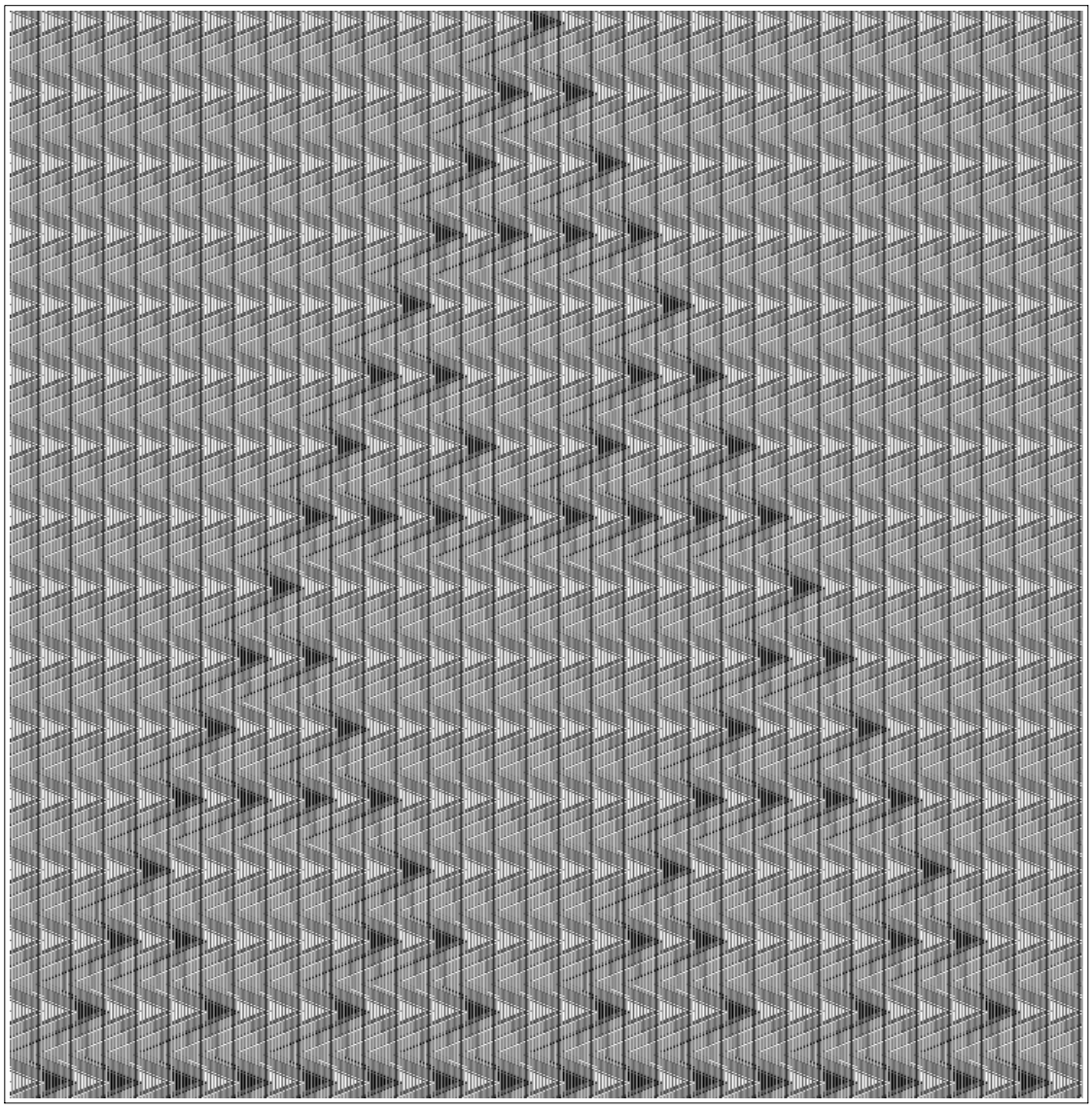




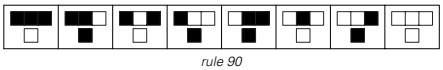
The universal cellular automaton emulating elementary rule 254. Each cell in rule 254 is represented by a block of 20 cells in the universal cellular automaton. Each of these blocks encodes both the color of the cell it represents, and the rule for updating this color.



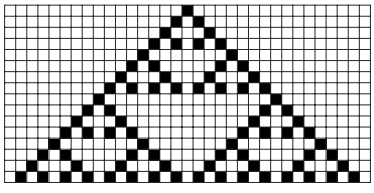
rule 254

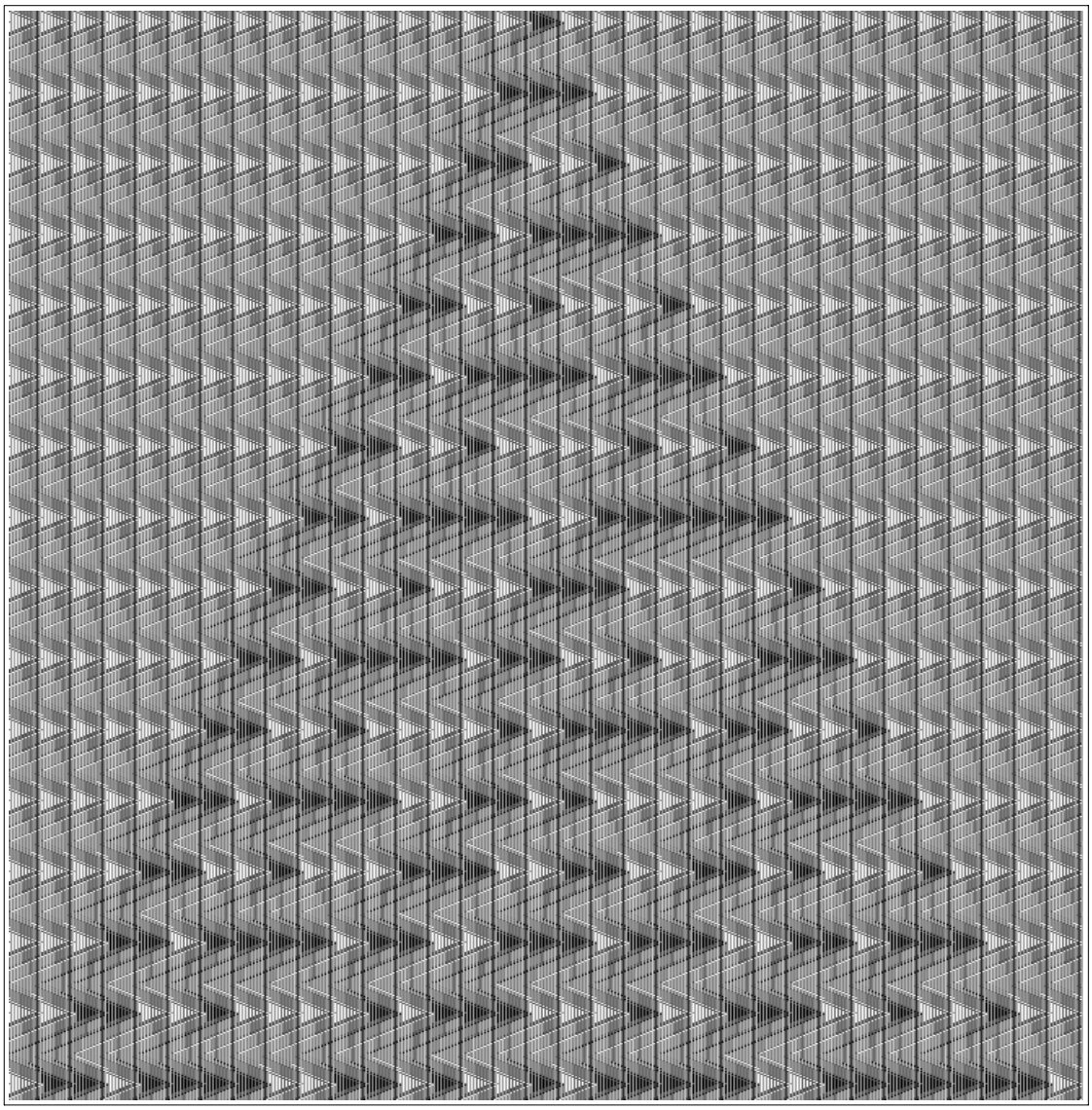


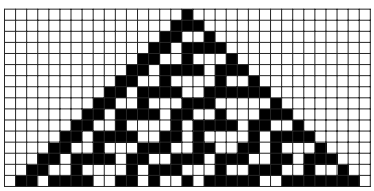
The universal cellular automaton emulating elementary rule 90. The underlying rules for the universal cellular automaton are exactly the same as on the previous page. But each block in the initial conditions now contains a representation of rule 90 rather than rule 254.



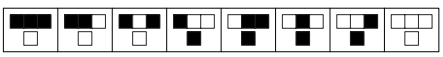
rule 90







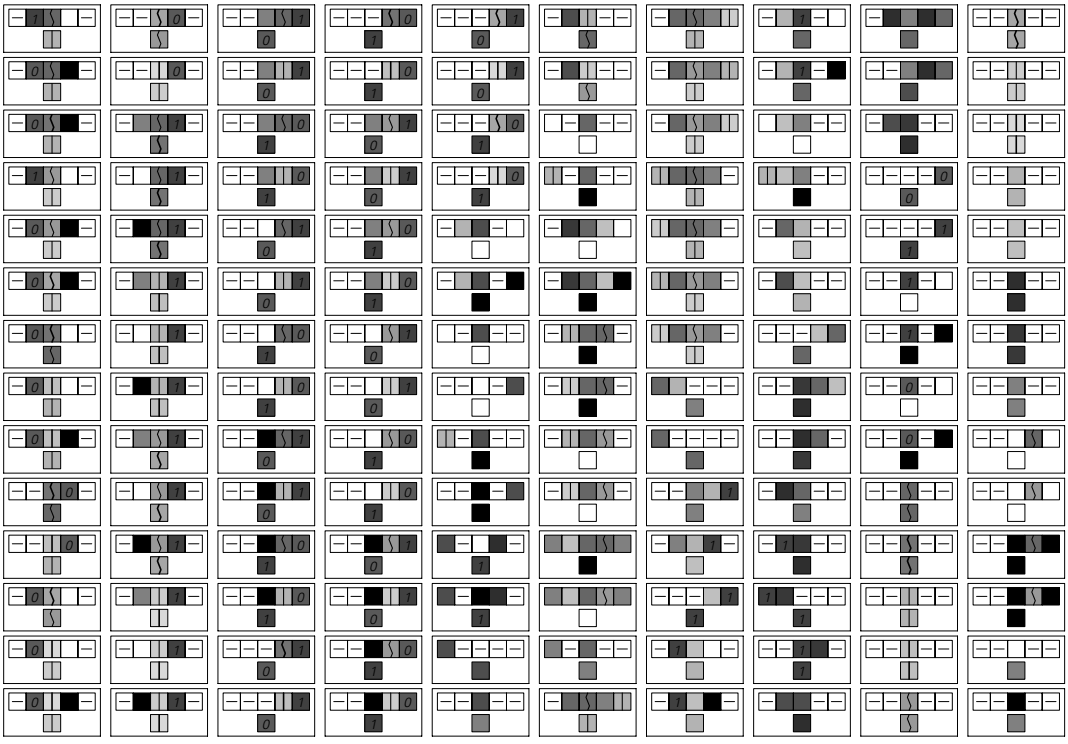
The universal cellular automaton emulating rule 30. A total of 848 steps in the evolution of the universal cellular automaton are shown, corresponding to 16 steps in the evolution of rule 30.



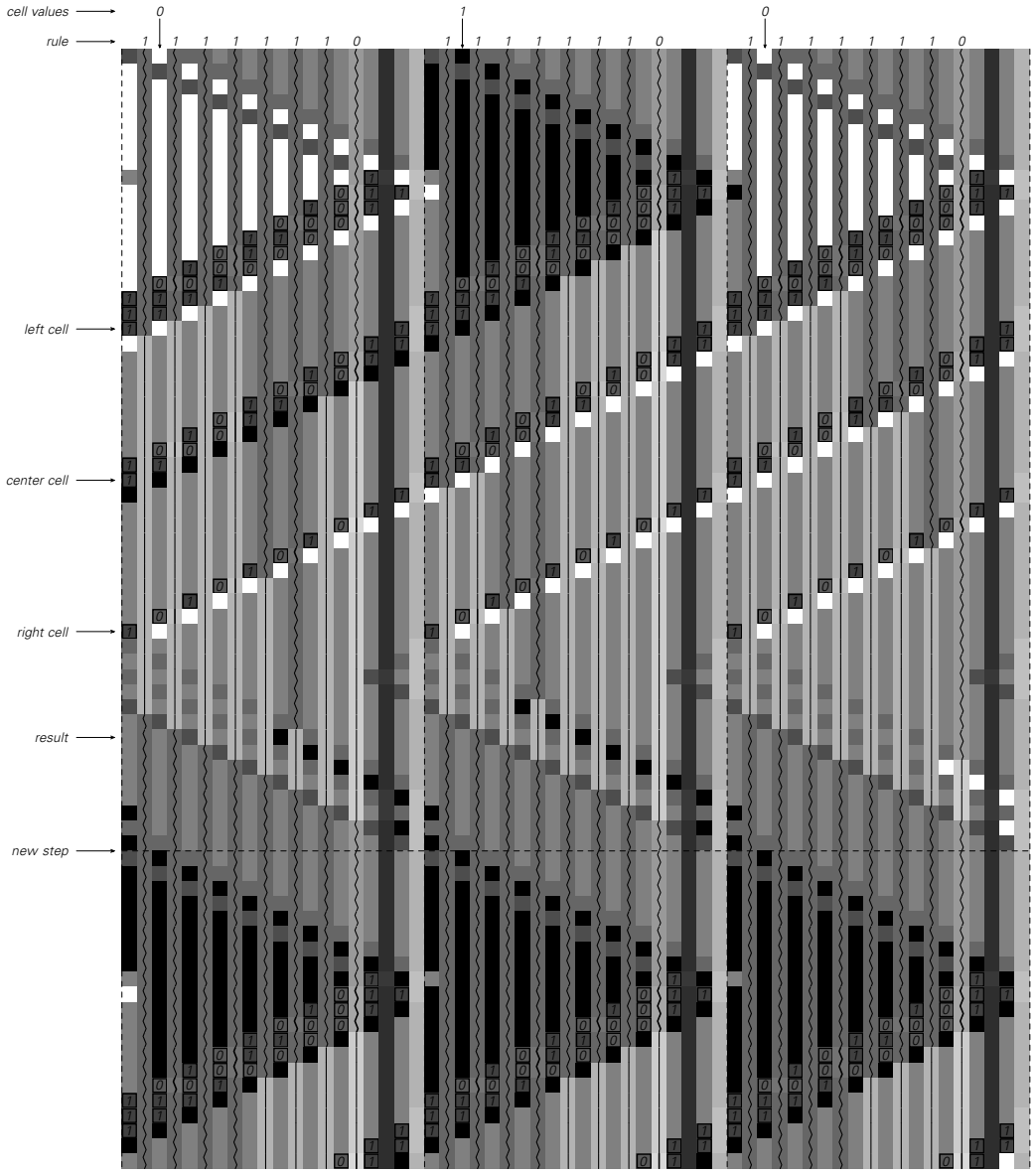
rule 30

On each page the underlying rules for the universal cellular automaton are exactly the same. But on the first page, the initial conditions are set up so as to make the universal cellular automaton emulate rule 254, while on the second page they are set up to make it emulate rule 90, and on the third page rule 30.

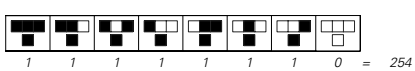
The pages that follow show how this works. The basic idea is that a block of 20 cells in the universal cellular automaton is used to represent each single cell in the cellular automaton that is being emulated. And within this block of 20 cells is encoded both a specification of the current color of the cell that is being represented, as well as the rule by which that color is to be updated.

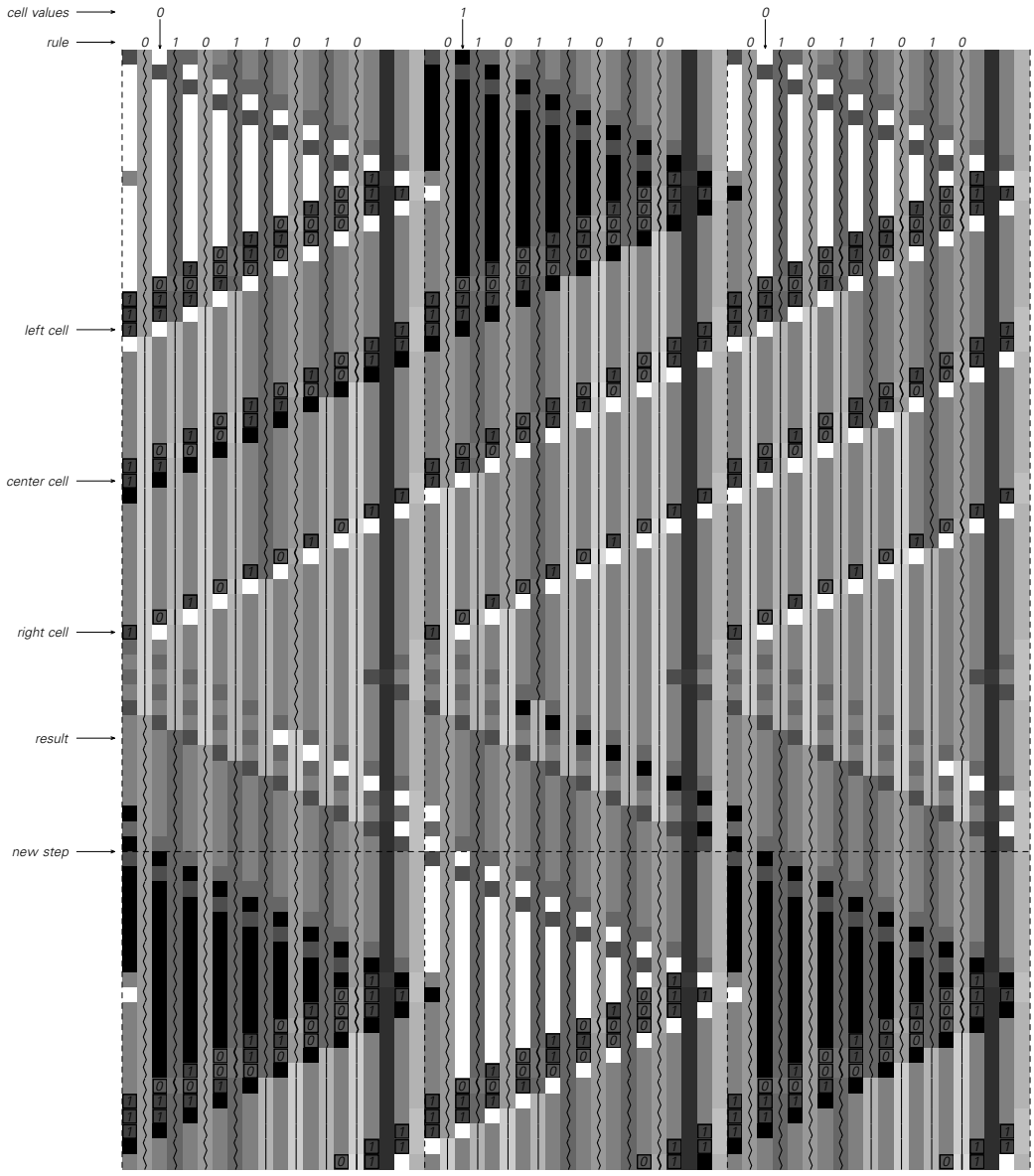


The rules for the universal cellular automaton. There are 19 possible colors for each cell, represented here by 19 different icons. Since the new color of each cell depends on the previous colors of a total of five cells, there are in principle 2,476,099 cases to cover. But by using ⊟ to stand for a cell with any possible color, many cases are combined. Note that the cases shown are in a definite order reading down successive columns, with special cases given before more general ones. With the initial conditions used, there are some combinations of cells that can never occur, and these are not covered in the rules shown.

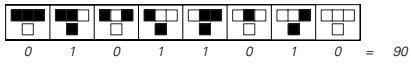


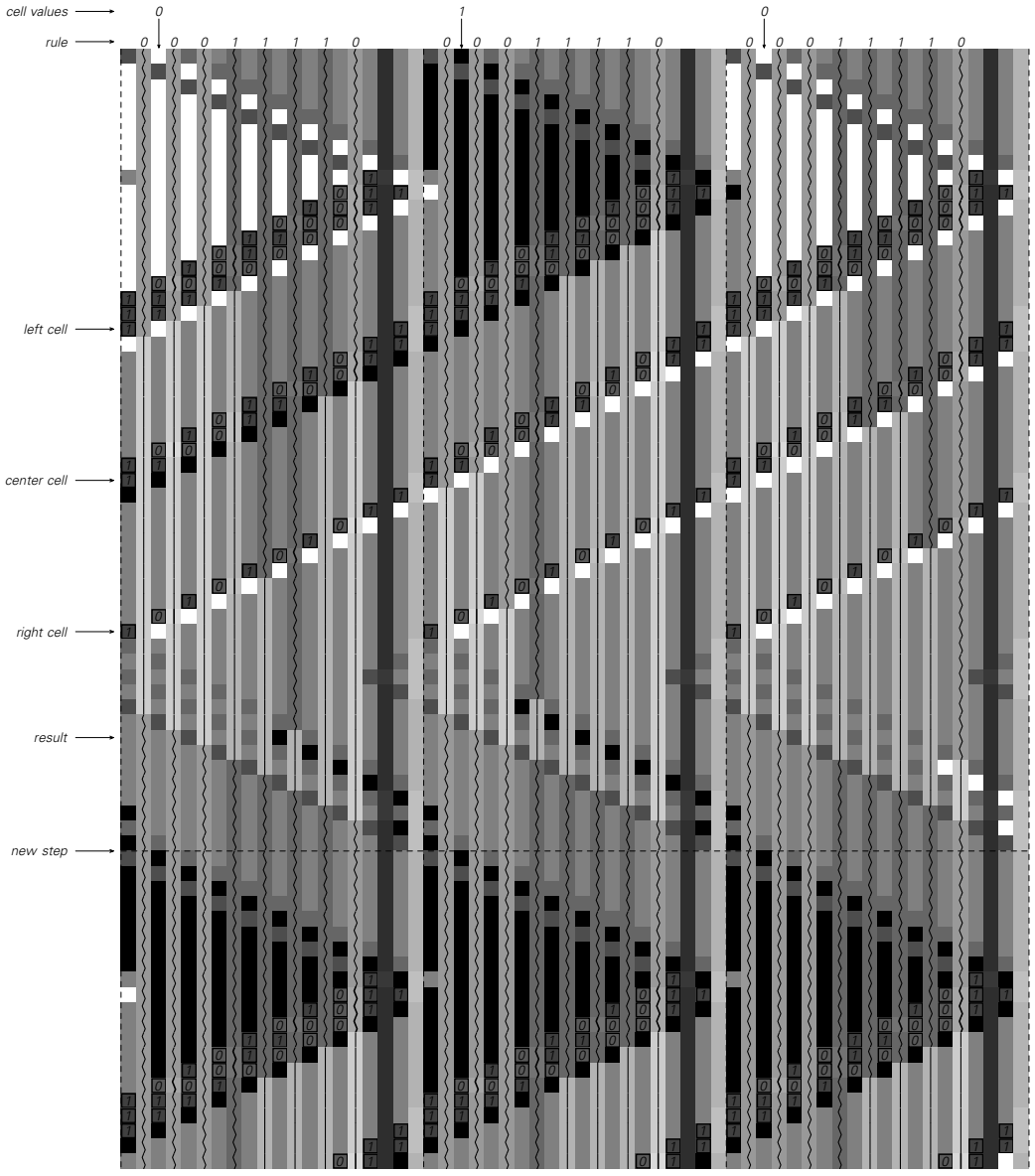
Details of how the universal cellular automaton emulates rule 254. Each of the blocks in the universal cellular automaton represents a single cell in rule 254, and encodes both the current color of the cell and the form of the rule used to update it.



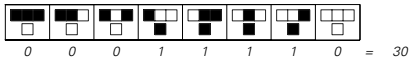


Details of how the universal cellular automaton emulates rule 90. The only difference in initial conditions from the picture on the previous page is that each block now encodes rule 90 instead of rule 254.





Details of how the universal cellular automaton emulates rule 30. Once again, the only difference in initial conditions from the facing page is that each block now encodes rule 30 instead of rule 90.



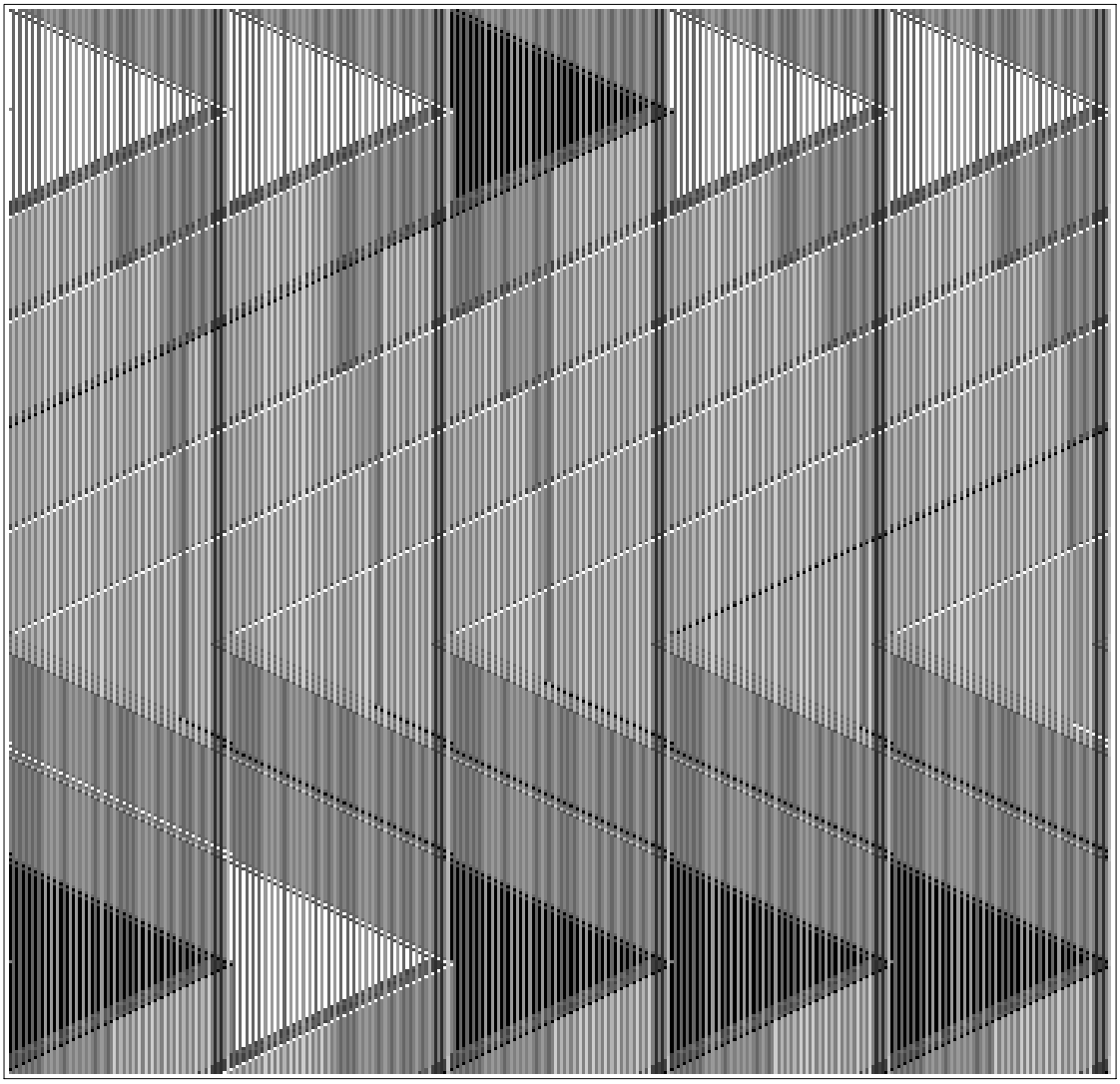
In the examples shown, the cellular automata being emulated have 8 cases in their rules, with each case giving the outcome for one of the 8 possible combinations of colors of a cell and its immediate neighbors. In every block of 20 cells in the universal cellular automaton, these rules are encoded in a very straightforward way, by listing in order the outcomes for each of the 8 possible cases.

To update the color of the cell represented by a particular block, what the universal cellular automaton must then do is to determine which of the 8 cases applies to that cell. And it does this by successively eliminating cases that do not apply, until eventually only one case remains. This process of elimination can be seen quite directly in the pictures on the previous pages. Below each large black or white triangle, there are initially 8 vertical dark lines. Each of these lines corresponds to one of the 8 cases in the rule, and the system is set up so that a particular line ends as soon as the case to which it corresponds has been eliminated.

It so happens that in the universal cellular automaton discussed here the elimination process for a given cell always occurs in the block immediately to the left of the one that represents that cell. But the process itself is not too difficult to understand, and indeed it works in much the way one might expect of a practical electronic logic circuit.

There are three basic stages, visible in the pictures as three stripes moving to the left across each block. The first stripe carries the color of the left-hand neighbor, and causes all cases in the rule where that neighbor does not have the appropriate color to be eliminated. The next two stripes then carry the color of the cell itself and of its right-hand neighbor. And after all three stripes have passed, only one of the 8 cases ever survives, and this case is then the one that gives the new color for the cell.

The pictures on the last few pages have shown how the universal cellular automaton can in effect be programmed to emulate any cellular automaton whose rules involve nearest neighbors and two possible colors for each cell. But the universal cellular automaton is in no way restricted to emulating only rules that involve nearest neighbors. And thus on the facing page, for example, it is shown emulating a rule that involves next-nearest as well as nearest neighbors.



The universal cellular automaton emulating one step in the evolution of the rule shown above, which involves next-nearest as well as nearest-neighbor cells. The rule now covers a total of 32 cases, corresponding to the possible arrangements of colors of a cell and its nearest and next-nearest neighbors. The picture shows the evolution of five cells according to the rule shown, with each cell now being represented by a block of 70 cells in the universal cellular automaton.

The blocks needed to represent each cell are now larger, since they must include all 32 cases in the rule. There are also five elimination stages rather than three. But despite these differences, the underlying rule for the universal cellular automaton remains exactly the same.

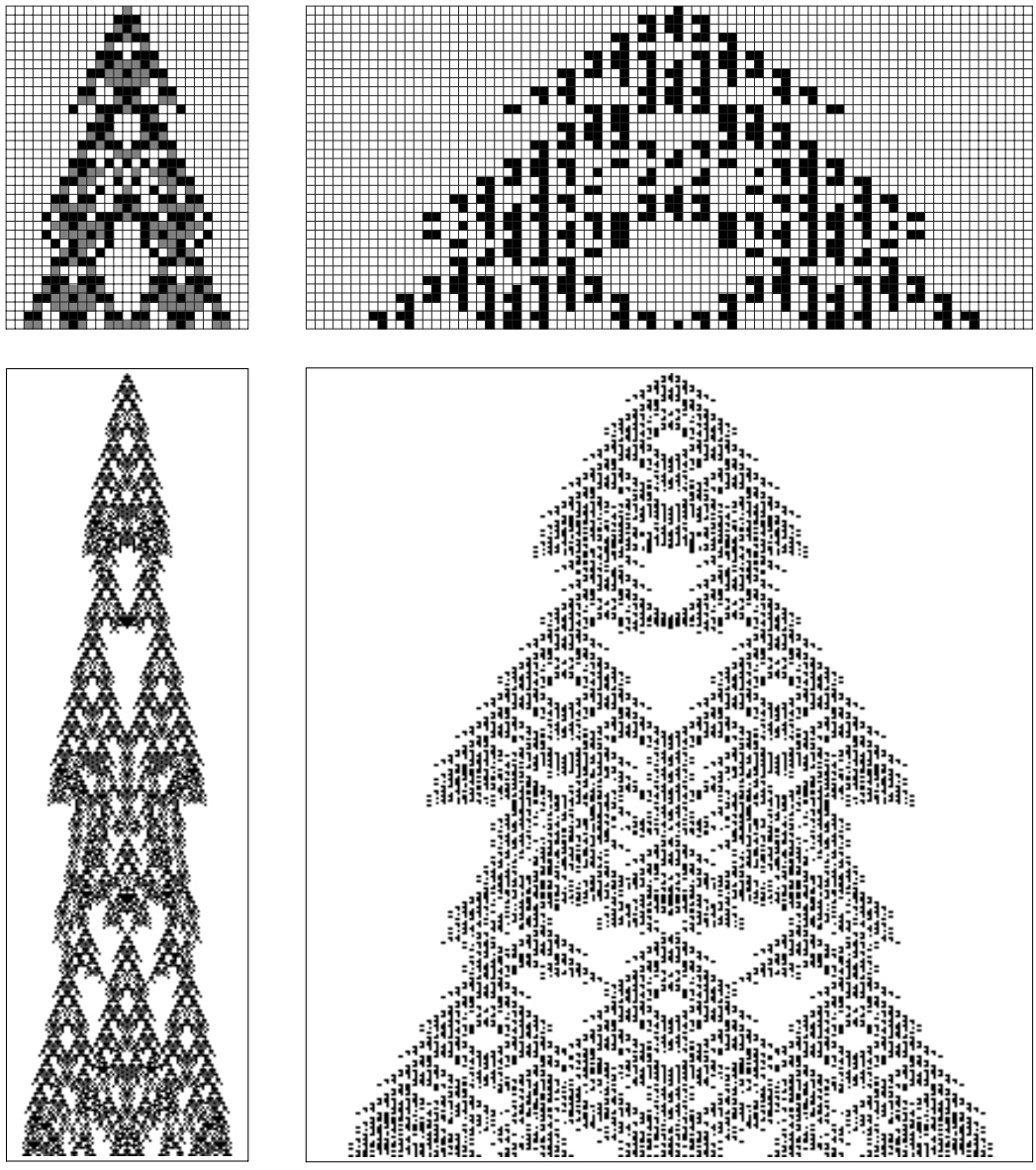
What about rules that have more than two possible colors for each cell? It turns out that there is a general way of emulating such rules by using rules that have just two colors but a larger number of neighbors. The picture on the facing page shows an example. The idea is that each cell in the three-color cellular automaton is represented by a block of three cells in the two-color cellular automaton. And by looking at neighbors out to distance five on each side, the two-color cellular automaton can update these blocks at each step in direct correspondence with the rules of the three-color cellular automaton.

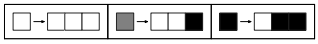
The same basic scheme can be used for rules with any number of colors. And the conclusion is therefore that the universal cellular automaton can ultimately emulate a cellular automaton with absolutely any set of rules, regardless of how many neighbors and how many colors they may involve.

This is an important and at first surprising result. For among other things, it implies that the universal cellular automaton can emulate cellular automata whose rules are more complicated than its own. If one did not know about the basic phenomenon of universality, then one would most likely assume that by using more complicated rules one would always be able to produce new and different kinds of behavior.

But from studying the universal cellular automaton in this section, we now know that this is not in fact the case. For given the universal cellular automaton, it is always in effect possible to program this cellular automaton to emulate any other cellular automaton, and therefore to produce whatever behavior the other cellular automaton could produce.

In a sense, therefore, what we can now see is that nothing fundamental can ever be gained by using rules that are more complicated than those for the universal cellular automaton. For given the universal cellular automaton, more complicated rules can always be emulated just by setting up appropriate initial conditions.





An example of how a cellular automaton with three possible colors and nearest-neighbor rules can be emulated by a cellular automaton with only two possible colors but a larger number of neighbors (in this case five on each side). The basic idea is to represent each cell in the three-color rule by a block of three cells in the two-color rule, according to the correspondence given on the left. The three-color rule illustrated here is totalistic code 1599 from page 70.

Looking at the specific universal cellular automaton that we have discussed in this section, however, we would probably be led to assume that while the phenomenon of universality might be important in principle, it would rarely be relevant in practice. For the rules of the universal cellular automaton in this section are quite complicated—involving 19 possible colors for each cell, and next-nearest as well as nearest neighbors. And if such complication was indeed necessary in order to achieve universality, then one would not expect that universality would be common, for example, in the systems we see in nature.

But what we will discover later in this chapter is that such complication in underlying rules is in fact not needed. Indeed, in the end we will see that universality can actually occur in cellular automata with just two colors and nearest neighbors. The operation of such cellular automata is considerably more difficult to follow than the operation of the universal cellular automaton discussed in this section. But the existence of universal cellular automata with such simple underlying rules makes it clear that the basic results we have obtained in this section are potentially of very broad significance.

Emulating Other Systems with Cellular Automata

The previous section showed that a particular universal cellular automaton could emulate any possible cellular automaton. But what about other types of systems? Can cellular automata also emulate these?

With their simple and rather specific underlying structure one might think that cellular automata would never be capable of emulating a very wide range of other systems. But what I will show in this section is that in fact this is not the case, and that in the end cellular automata can actually be made to emulate almost every single type of system that we have discussed in this book.

As a first example of this, the picture on the facing page shows how a cellular automaton can be made to emulate a mobile automaton.

The main difference between a mobile automaton and a cellular automaton is that in a mobile automaton there is a special active cell that moves around from one step to the next, while in a cellular