EXCERPTED FROM

STEPHEN WOLFRAM
A NEW KIND OF SCIENCE

SECTION 11.5

Emulating Other Systems with Cellular Automata

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.

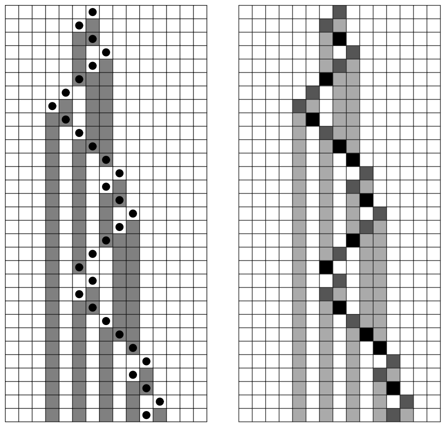
Emulating Other Systems with Cellular Automata

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?

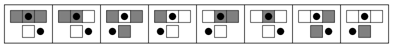
With their simple and rather specific underlying structure one might think that cellular automata would never be capable of emulating a very wide range of other systems. But what I will show in this section is that in fact this is not the case, and that in the end cellular automata can actually be made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

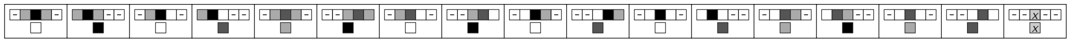
As a first example of this, the picture on the facing page shows how a cellular automaton can be made to emulate a mobile automaton.

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



An example of a mobile automaton (see page 71) being emulated by a cellular automaton. In the mobile automaton shown on the left each cell has two possible colors. In the cellular automaton shown on the right, the cells have four possible colors, with two darker colors corresponding to the active cell in the mobile automaton. The rules for the mobile automaton and the cellular automaton are shown below. In the rules for the cellular automaton, ⊟ indicates a cell of any color.



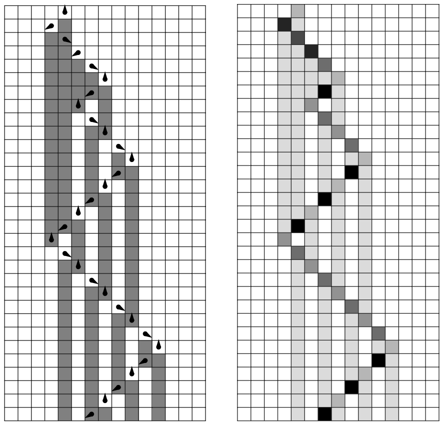


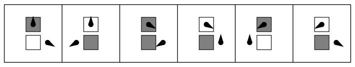
automaton all cells are always effectively treated as being exactly the same. And to emulate a mobile automaton with a cellular automaton it turns out that all one need do is to divide the possible colors of cells in the cellular automaton into two sets: lighter ones that correspond to ordinary cells in the mobile automaton, and darker ones that correspond to active cells. And then by setting up appropriate rules and choosing initial conditions that contain only one darker cell, one can produce in the cellular automaton an exact emulation of every step in the evolution of a mobile automaton—as in the picture above.

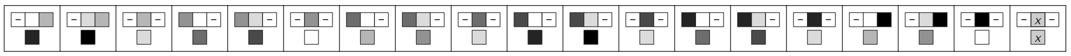
The same basic approach can be used to construct a cellular automaton that emulates a Turing machine, as illustrated on the next page. Once again, lighter colors in the cellular automaton represent ordinary cells in the Turing machine, while darker colors represent the cell under the head, with a specific darker color corresponding to each possible state of the head.

One might think that the reason that mobile automata and Turing machines can be emulated by cellular automata is that they both consist of fixed arrays of cells, just like cellular automata. So then one may wonder what happens with substitution systems, for example, where there is no fixed array of elements.

An example of a Turing machine being emulated by a cellular automaton. In the Turing machine on the left each cell has two possible colors, and the head has three possible states. In the cellular automaton, the cells have eight possible colors, with the lightest two colors being used for cells not at the position of the head. The rules for the Turing machine and the cellular automaton are shown below. In the rules for the cellular automaton, ⊟ indicates a cell of any color.





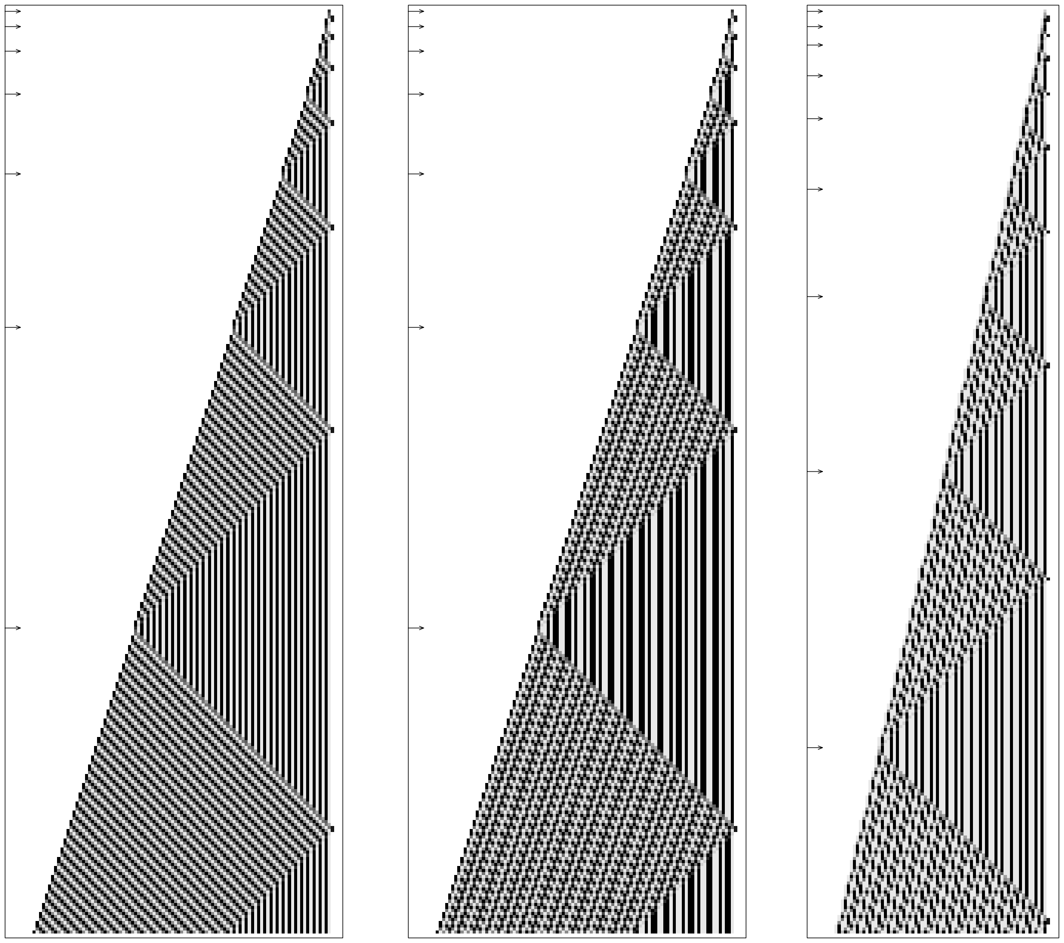


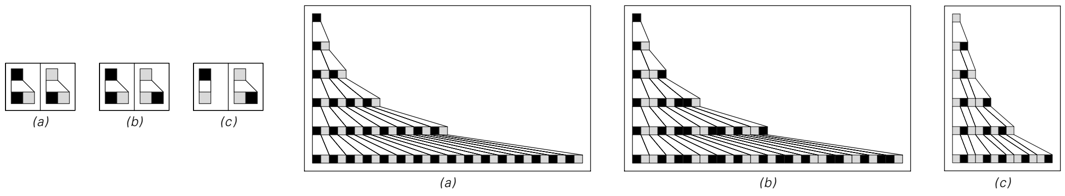
The pictures on the facing page demonstrate that in fact these can also be emulated by cellular automata. But while one can emulate each step in the evolution of a mobile automaton or a Turing machine with a single step of cellular automaton evolution, this is no longer in general true for substitution systems.

That this must ultimately be the case one can see from the fact that the total number of elements in a substitution system can be multiplied by a factor from one step to the next, while in a cellular automaton the size of a pattern can only ever increase by a fixed amount at each step. And what this means is that it can take progressively larger numbers of cellular automaton steps to reproduce each successive step in the evolution of the substitution system—as illustrated in the pictures on the facing page.

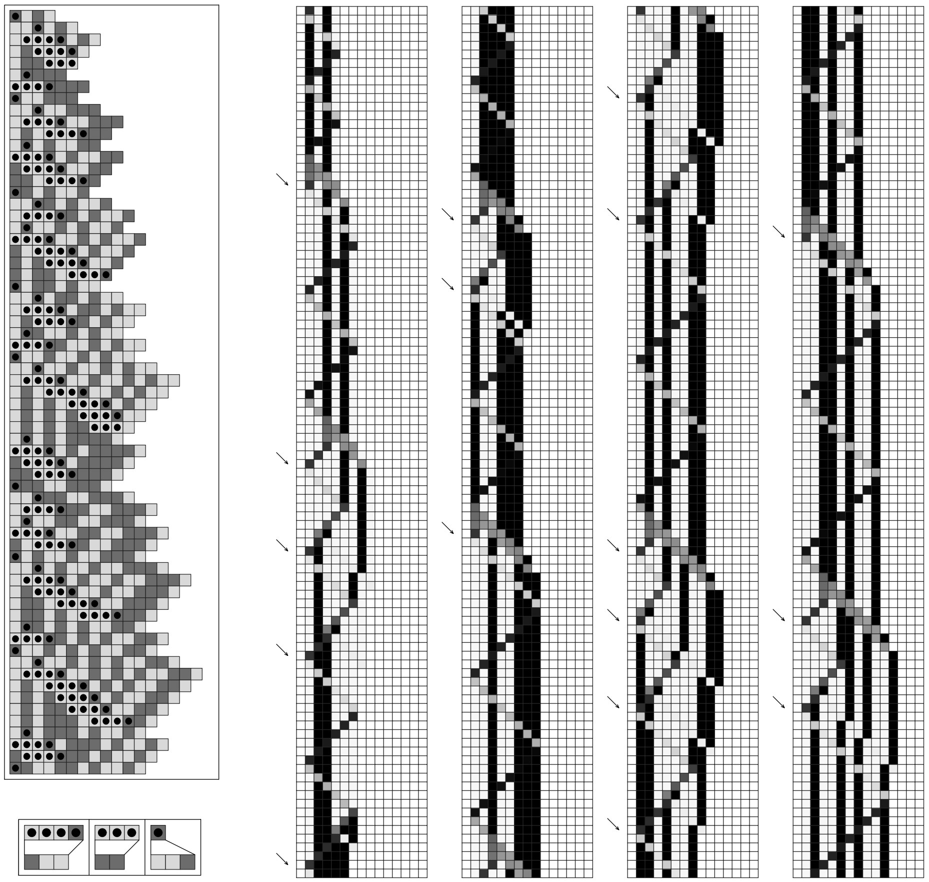
The same kind of problem occurs in sequential substitution systems—as well as in tag systems. But once again, as the pictures on page 660 demonstrate, it is still perfectly possible to emulate systems like these using cellular automata.

But just how broad is the set of systems that cellular automata can ultimately emulate? All the examples of systems that I have shown so far can at some level be thought of as involving sequences of elements that are fairly directly analogous to the cells in a cellular automaton.



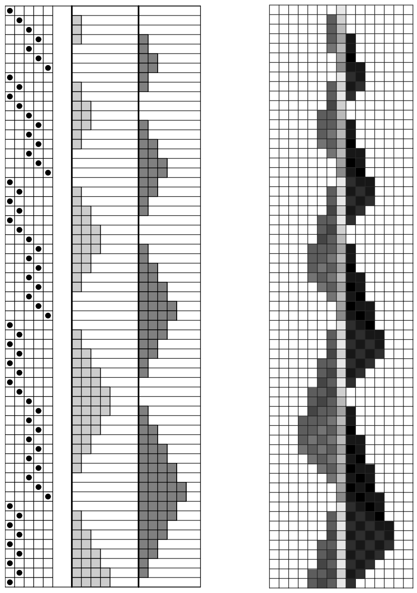


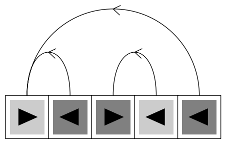
Examples of cellular automata that emulate substitution systems. The successive steps in the evolution of each substitution system are obtained at the points indicated by arrows. Note that the sequences of elements generated by the cellular automata are aligned at the right, while in the pictures of the substitution systems shown they are aligned at the left. The rules for the three cellular automata involve only nearest neighbors, and allow 12 possible colors for each cell.


A cellular automaton set up to emulate a sequential substitution system. The cellular automaton involves 28 colors and nearest-neighbor rules. The strings produced by the sequential substitution system appear on successive diagonal stripes indicated by arrows in the evolution of the cellular automaton on the right.

But one example where there is no such direct analogy is a register machine. And at the outset one might not imagine that such a system could ever readily be emulated by a cellular automaton.

But in fact it turns out to be fairly straightforward to do so, as illustrated at the top of the facing page. The basic idea is to have the cellular automaton produce a pattern that expands and contracts on each side in a way that corresponds to the incrementing and decrementing of the sizes of numbers in the first and second registers of



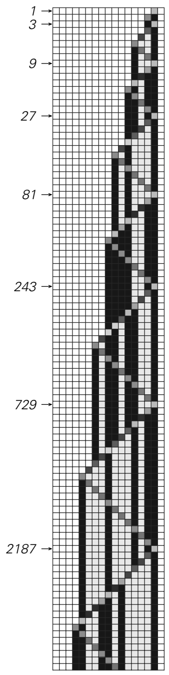


An example of a register machine being emulated by a cellular automaton. The cellular automaton has 12 possible colors for each cell. Of these, 5 are used by the center cell to represent the point that has been reached in the register machine program. The other 7 are used to implement signals that propagate out to the left and right to do the analog of incrementing and decrementing each register.

the register machine. In the center of the cellular automaton is then a cell whose possible colors correspond to possible points in the program for the register machine. And as the cell makes transitions from one color to another, it effectively emits signals that move to the left or right modifying the pattern in the cellular automaton in a way that follows each instruction in the register machine program.

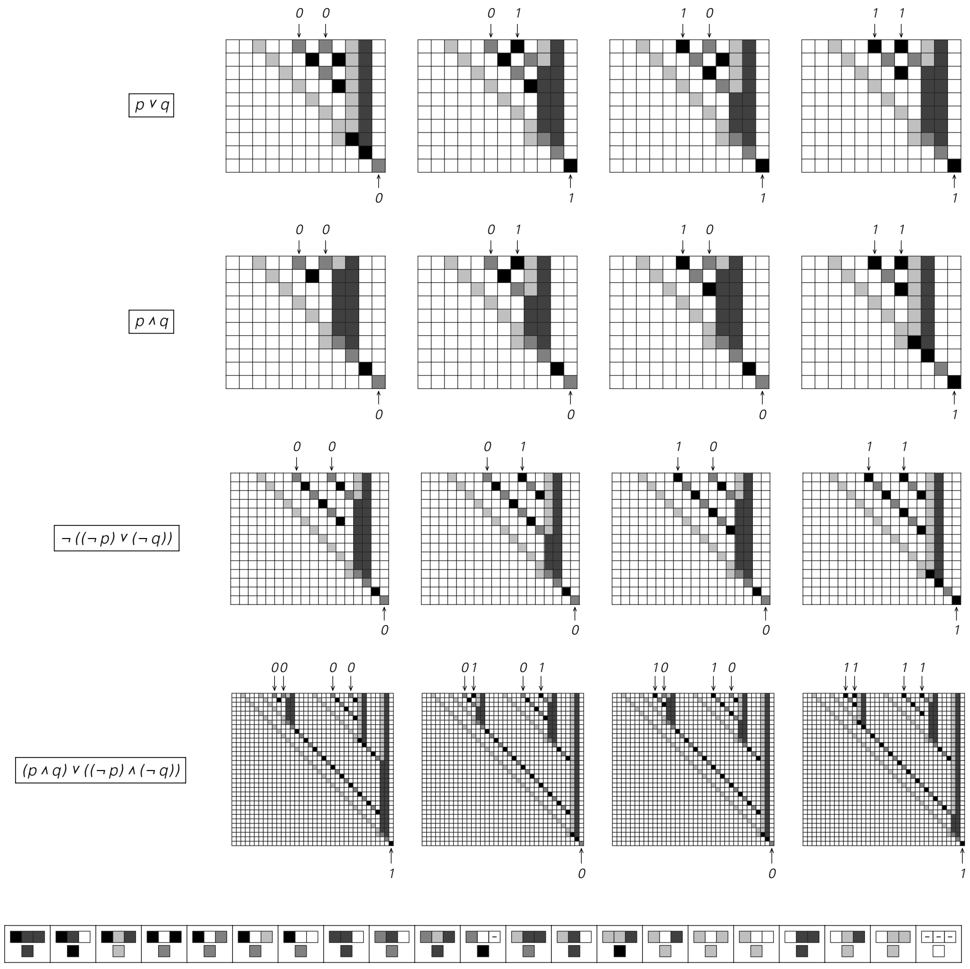
So what about systems based on numbers? Can these also be emulated by cellular automata? As one example the picture on the right shows how a cellular automaton can be set up to perform repeated multiplication by 3 of numbers in base 2. And the only real difficulty in this case is that carries generated in the process of multiplication may need to be propagated from one end of the number to the other.

So what about practical computers? Can these also be emulated by cellular automata? From the examples just discussed of register machines and systems based on numbers, we already know that cellular automata can emulate some of the low-level operations typically found in computers. And the pictures on the next two pages show how cellular automata can also be made to emulate two other important aspects of practical computers.

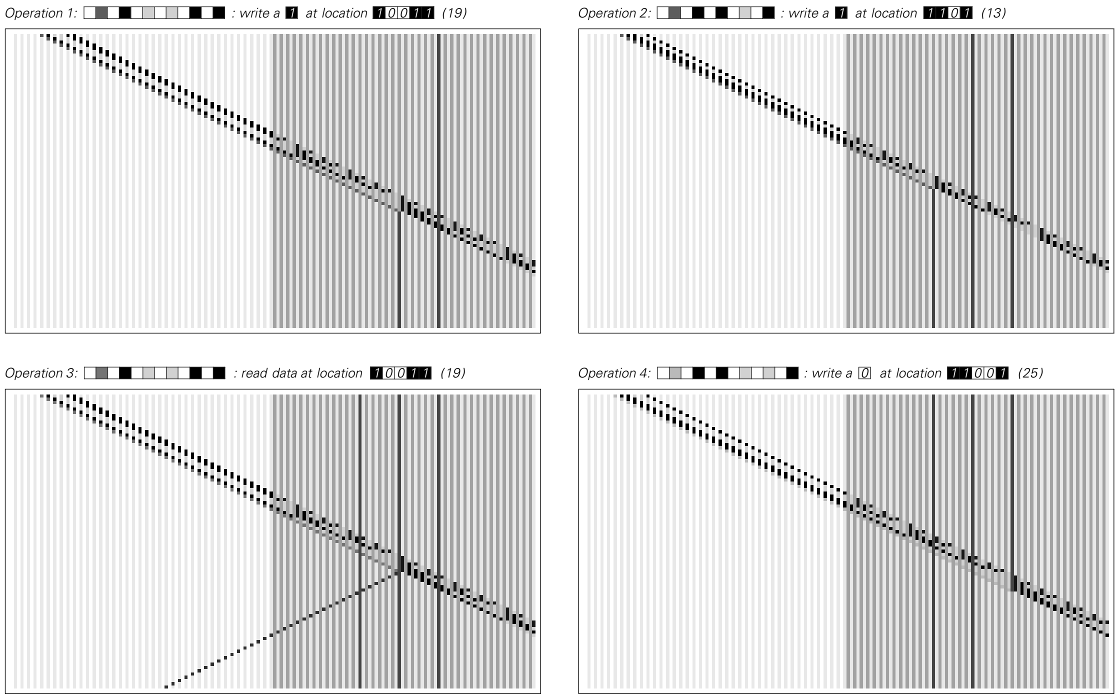


Repeated multiplication by 3 in base 2 being performed by a cellular automaton with 11 colors.

The pictures below show how a cellular automaton can evaluate any logic expression that is given in a certain form. And the picture on the facing page then shows how a cellular automaton can retrieve data from a numbered location in what is effectively a random-access memory.



A cellular automaton which emulates basic logic circuits. The underlying rules for the cellular automaton are exactly the same in each case, and involve nearest neighbors and five possible colors for each cell. But the initial condition can represent a logic expression that involves any number of variables together with the operations of AND, OR and NOT. In the examples above, two variables, *p* and *q*, are used, and in each case the behavior obtained with all four possible combinations of values for *p* and *q* are shown.


Operation 1: : write a 1 at location 10011 (19)
Operation 2: : write a 1 at location 1101 (13)
Operation 3: : read data at location 10011 (19)
Operation 4: : write a 0 at location 11001 (25)

A cellular automaton set up to emulate random-access memory in a computer. The memory is on the right, and can be of any size. Instructions come in from the left, with memory locations specified by addresses consisting of binary digits.

The details for any particular case are quite complicated, but in the end it turns out that it is in principle possible to construct a cellular automaton that emulates a practical computer in its entirety.

And as a result, one can conclude that any of the very wide range of computations that can be performed by practical computers can also be done by cellular automata.

From the previous section we know that any cellular automaton can be emulated by a universal cellular automaton. But now we see that a universal cellular automaton is actually much more universal than we saw in the previous section. For not only can it emulate any cellular automaton: it can also emulate any of a wide range of other systems, including practical computers.