EXCERPTED FROM

STEPHEN WOLFRAM
A NEW KIND OF SCIENCE

SECTION 10.11

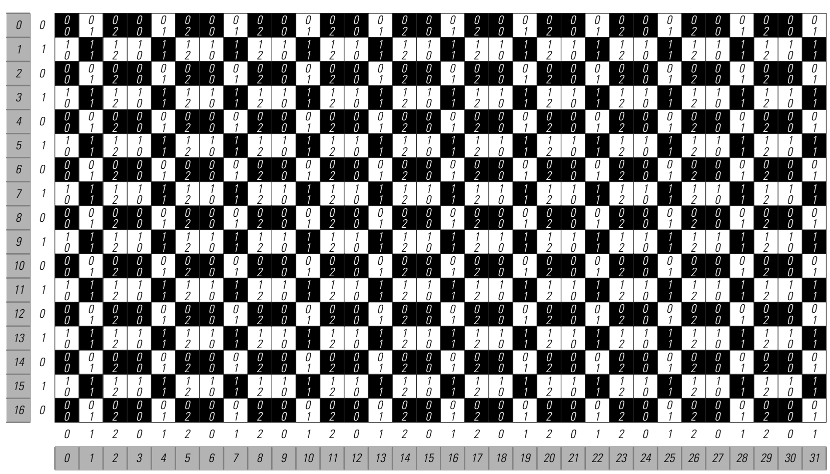
Traditional Mathematics and Mathematical Formulas

complicated—so that the approach probably ends up essentially being no better than just enumerating possible initial conditions.

The conclusion therefore is that at least with standard methods of cryptanalysis—as well as a few others—there appears to be no easy way to deduce the key for rule 30 from any suitably chosen encrypting sequence. But how can one be sure that there really is absolutely no easy way to do this? In Chapter 12 I will discuss some fundamental approaches to such a question. But as a practical matter one can say that not only have direct attempts to find easy ways to deduce the key in rule 30 failed, but also—despite some considerable effort—little progress has been made in solving any of various problems that turn out to be equivalent to this one.

Traditional Mathematics and Mathematical Formulas

Traditional mathematics has for a long time been the primary method of analysis used throughout the theoretical sciences. Its goal can usually be thought of as trying to find a mathematical formula that summarizes the behavior of a system. So in a simple case if one has an array of black and white squares, what one would typically look for is a formula that takes the numbers which specify the position of a particular square and from these tells one whether the square is black or white.



With a pattern that is purely repetitive, the formula is always straightforward, as the picture at the bottom of the facing page illustrates. For all one ever need do is to work out the remainder from dividing the position of a particular square by the size of the basic repeating block, and this then immediately tells one how to look up the color one wants.

So what about nested patterns? It turns out that in most of traditional mathematics such patterns are already viewed as quite advanced. But with the right approach, it is in the end still fairly straightforward to find formulas for them.

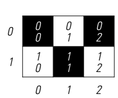
The crucial idea—much as in Chapter 4—is to think about numbers not in terms of their size but instead in terms of their digit sequences. And with this idea the picture on the next page shows an example of how what is in effect a formula can be constructed for a nested pattern.

What one does is to look at the digit sequences for the numbers that give the vertical and horizontal positions of a certain square. And then in the specific case shown one compares corresponding digits in these two sequences, and if these digits are ever respectively 0 and 1, then the square is white; otherwise it is black.

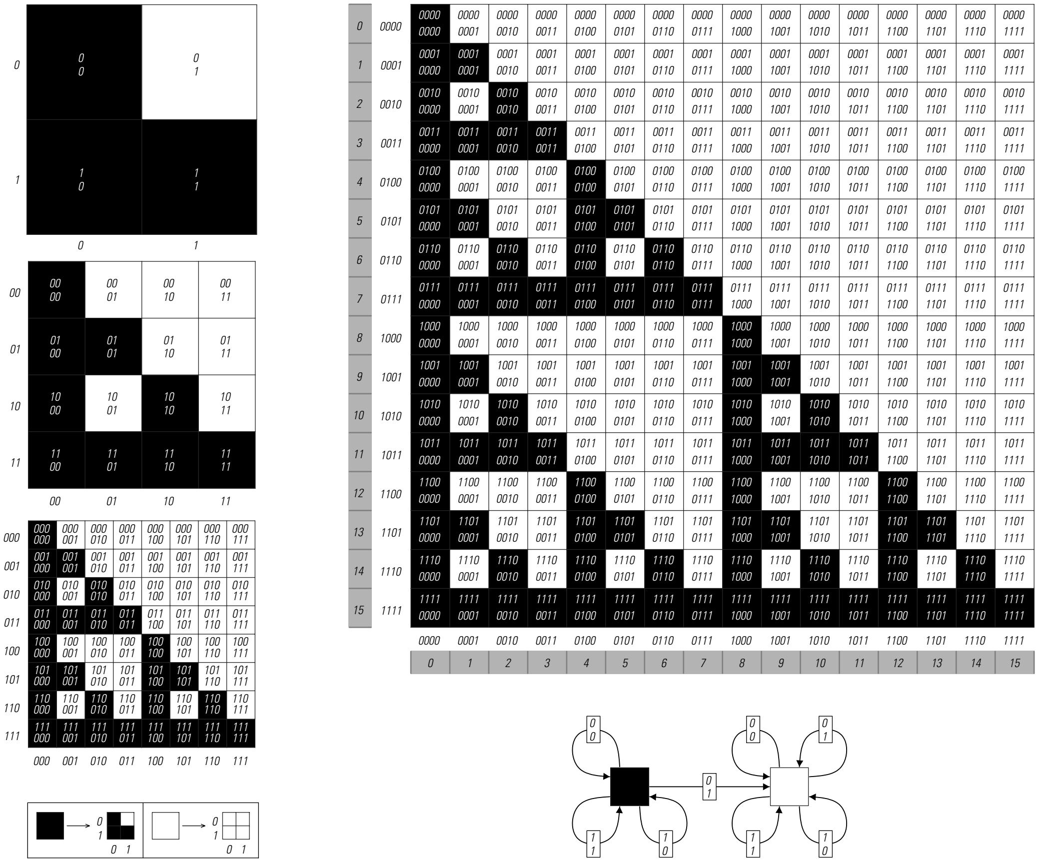
So why does this procedure work?

As we have discussed several times in this book, any nested pattern must—almost by definition—be able to be reproduced by a neighbor-independent substitution system. And in the case shown on the next page the rules for this system are such that they replace each square at each step by a 2×2 block of new squares. So as the picture illustrates this means that new squares always have positions that involve numbers containing one extra digit. With the particular rules shown, the new squares always have the same color as the old one, except in one specific case: when a black square is replaced, the new square that appears in the upper right is always white. But this square

◀ An example of how the color of any square in a repetitive pattern can be found from its coordinates by a simple mathematical procedure. The procedure takes the *x* and *y* coordinates of the square, and computes their remainders after division by 3 and 2 respectively. Using these remainders—which are shown inside each square—the color of a particular square can be



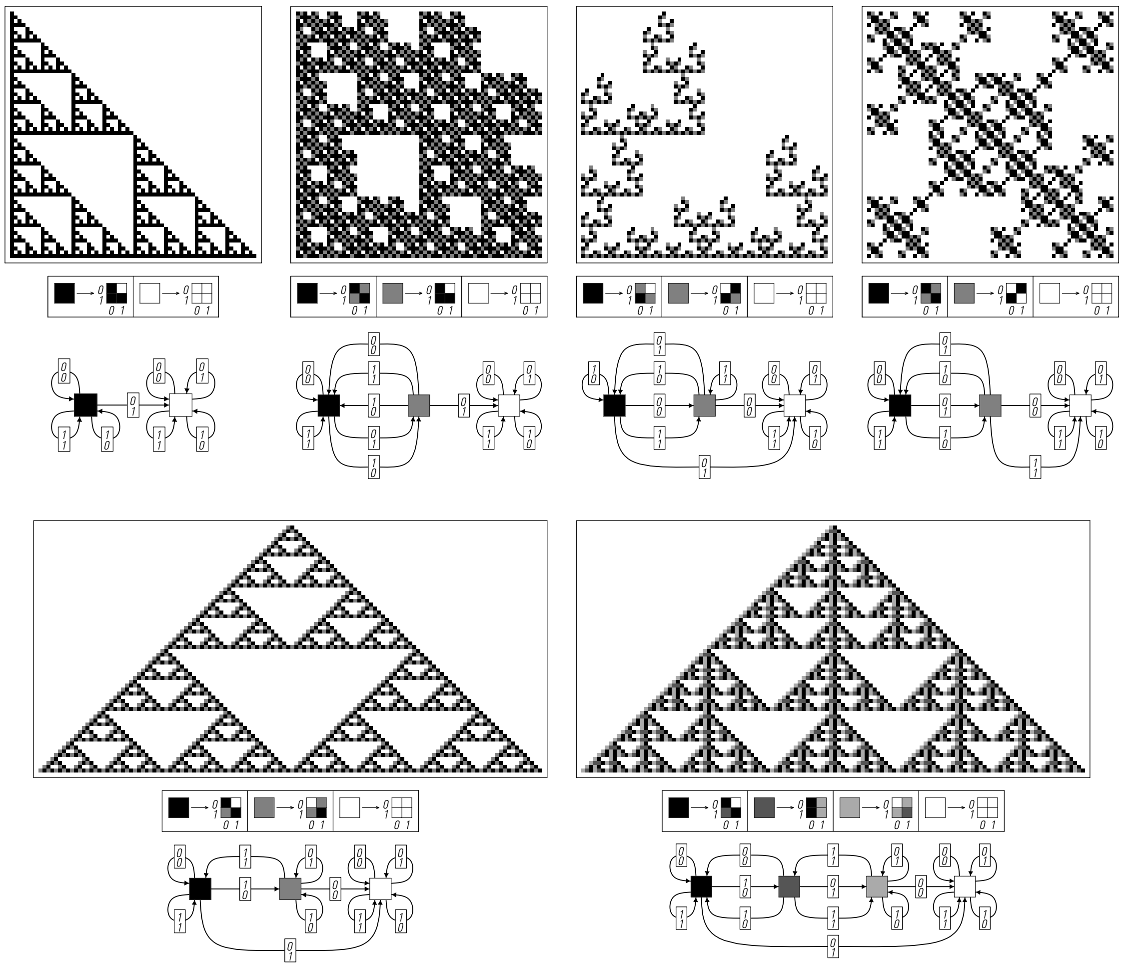
determined by a simple lookup in the repeating block shown on the left. The whole procedure can be represented using a mathematical formula that involves either functions like *Mod* or more traditional functions like *Sin*.



An example of how the color of any square in a nested pattern can be found from its coordinates by a fairly simple mathematical procedure. The procedure works by looking at the base 2 digit sequences of the coordinates. If any digit in the *y* coordinate of a particular square is 0 when the corresponding digit in the *x* coordinate is 1 then the square is white; otherwise it is black. The finite automaton at the bottom right gives a representation of this rule. Starting from the black square, one follows the sequence of connections that corresponds to the successive digits that one encounters in the *y* and *x* coordinates. Whatever square one lands up at in the finite automaton then gives the color one wants. Why this procedure works is illustrated by the pictures on the left. The nested pattern can be built up by a 2D substitution system with the rules shown. At each step in the evolution of this substitution system one gets a finer grid of squares, each specified in effect by one more digit in their coordinates.

has the property that its vertical position ends with a 0, and its horizontal position ends with a 1. So if the numbers that correspond to the position of a particular square contain this combination of digits at any point, it follows that the square must be white.

So what about other nested patterns? It turns out that using an extension of the argument above it is always possible to take the rules for the substitution system that generates a particular nested pattern, and from these construct a procedure for finding the color of a square in the pattern given its position. The pictures below show several examples, and in all cases the procedures are fairly straightforward.



Procedures for determining the color of a square at a given position in various nested patterns. In each case the whole pattern can be generated by repeatedly applying the substitution system rule shown. The color of any particular square can also be found by feeding the digit sequences of its y and x coordinates to the finite automaton shown. The first example shown corresponds to cellular automaton rule 60; the last two examples correspond respectively to rules 90 and 150. In the top row of examples, the initial condition for the substitution system is a single black square, and the start state for the finite automaton is also its black state. In the second row of examples, the initial condition consists of a light gray square next to a black square. In these cases, the colors of squares to the left of the center can be found by starting from the light gray state in the finite automaton; the colors of squares to the right can be found by starting from the black state.

But while these procedures could easily be implemented as programs, they are in a sense not based on what are traditionally thought of as ordinary mathematical functions. So is it in fact possible to get formulas for the colors of squares that involve only such functions?

In the one specific case shown at the top of the facing page it turns out to be fairly easy. For it so happens that this particular pattern—which is equivalent to the patterns at the beginning of each row on the previous page—can be obtained just by adding together pairs of numbers in the format of Pascal's triangle and then putting a black square whenever there is an entry that is an odd number.

And as the table below illustrates, the entries in Pascal's triangle are simply the binomial coefficients that appear when one expands out the powers of $1 + x$. So to determine whether a particular square in the pattern is black or white, all one need do is to compute the corresponding binomial coefficient, and see whether or not it is an odd number. And this means that if black is represented by 1 and white by 0, one can then give an explicit formula for the color of the square at position x on row y: it is simply (1 − (−1)^Binomial[y, x])/2.

1	1
$1+x$	$1+x$
$(1+x)^2$	$1+2x+x^2$
$(1+x)^3$	$1+3x+3x^2+x^3$
$(1+x)^4$	$1+4x+6x^2+4x^3+x^4$
$(1+x)^5$	$1+5x+10x^2+10x^3+5x^4+x^5$

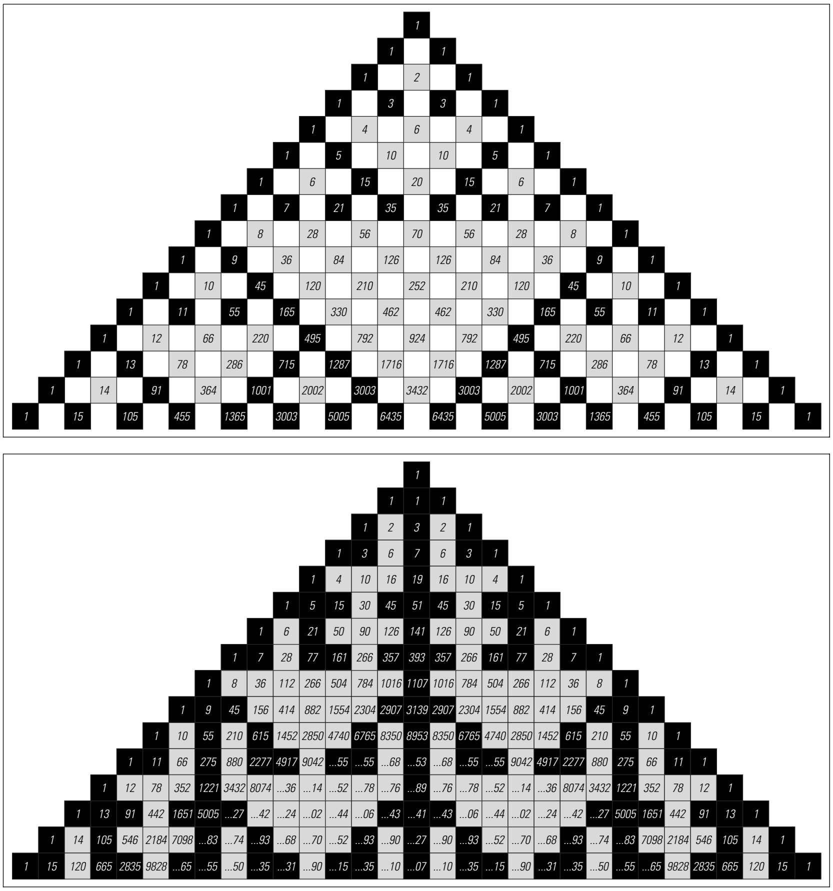
Binomial[t,n]

1	1
$1+x+x^2$	$1+x+x^2$
$(1+x+x^2)^2$	$1+2x+3x^2+2x^3+x^4$
$(1+x+x^2)^3$	$1+3x+6x^2+7x^3+6x^4+3x^5+x^6$
$(1+x+x^2)^4$	$1+4x+10x^2+16x^3+19x^4+16x^5+10x^6+4x^7+x^8$
$(1+x+x^2)^5$	$1+5x+15x^2+30x^3+45x^4+51x^5+45x^6+30x^7+15x^8+5x^9+x^{10}$

GegenbauerC[n,-t,-1/2]

Algebraic representations of the patterns on the facing page. The coefficient of x^n on each row gives the value of each square. These coefficients can also be obtained from the formulas in terms of *Binomial* and *GegenbauerC* given. A particular square is colored black if its value *a* is odd. This can be determined either from *Mod[a, 2]* or equivalently from *(1-(-1)^a)/2* or *Sin[π a/2]^2*. The succession of polynomials above can be obtained by expanding the generating functions *1/(1-(1+x) y)* and *1/(1-(1+x+x^2) y)*. *Binomial[m, n]* is the ordinary binomial coefficient *m!/(n! (m-n)!)*. *GegenbauerC* is a so-called orthogonal polynomial—a higher mathematical function.

So what about the bottom picture on the facing page? Much as in the top picture numbers can be assigned to each square, but now these numbers are computed by successively adding together triples rather



Nested patterns constructed using arithmetic operations. The example at the top is Pascal's triangle, formed by making each number be the sum of the numbers immediately to its left and right on the row above. The example at the bottom is a generalization of Pascal's triangle in which each number is the sum of the numbers above it and to its left and right on the row above. In both cases squares are colored black when the numbers that appear in them are odd. The limiting arrangements of colors correspond to nested patterns. For the top picture the pattern is what would be generated by an additive cellular automaton following rule 90; for the bottom picture it is what would be generated by one following rule 150. The numbers in the top picture are binomial coefficients; those in the bottom picture are particular trinomial coefficients.

than pairs. And once again the numbers appear as coefficients, but now in the expansion of powers of $1 + x + x^2$ rather than of $1 + x$.

So is there an explicit formula for these coefficients? If one restricts oneself to a fixed number of elementary mathematical functions together with factorials and multinomial coefficients then it appears that there is not. But if one also allows higher mathematical functions then it turns out that such a formula can in fact be found: as indicated in the table above each coefficient is given by a particular value of a so-called Gegenbauer or ultraspherical function.

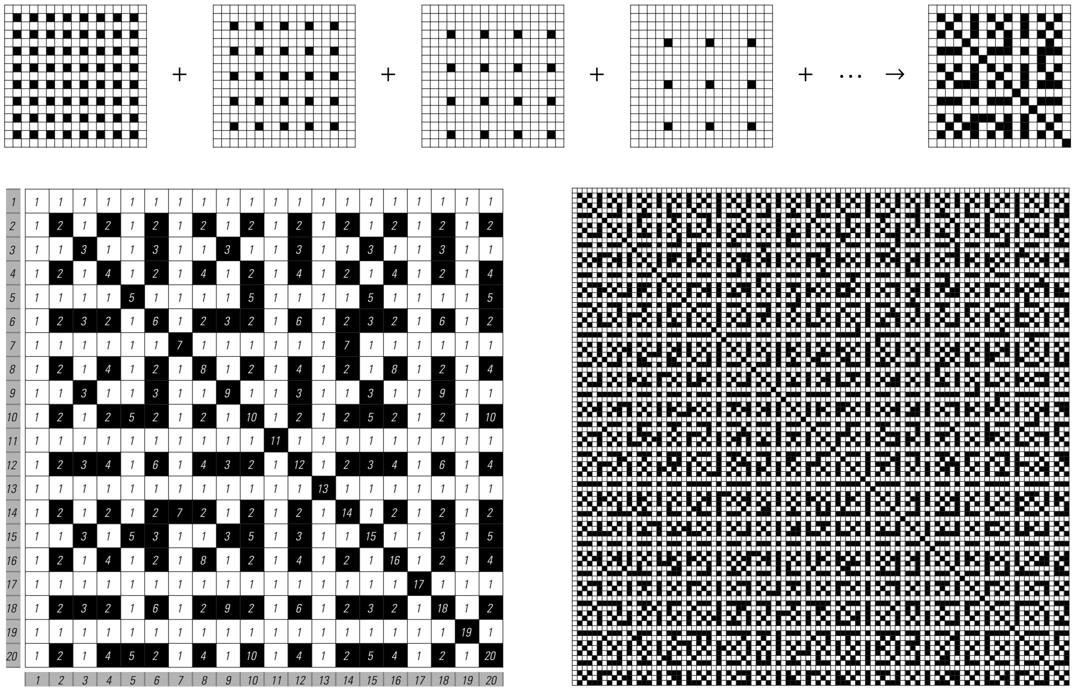
So what about other nested patterns? Both of the patterns shown on the previous page are rather special in that as well as being generated by substitution systems they can also be produced one row at a time by the evolution of one-dimensional cellular automata with simple additive rules. And in fact the approaches used above can be viewed as direct generalizations of such additive rules to the domain of ordinary numbers.

For a few other nested patterns there exist fairly simple connections with additive cellular automata and similar systems—though usually in more dimensions or with more neighbors. But for most nested patterns there seems to be no obvious way to relate them to ordinary mathematical functions. Nevertheless, despite this, it is my guess that in the end it will in fact turn out to be possible to get a formula for any nested pattern in terms of suitably generalized hypergeometric functions, or perhaps other functions that are direct generalizations of ones used in traditional mathematics.

Yet given how simple and regular nested patterns tend to look it may come as something of a surprise that it should be so difficult to represent them as traditional mathematical formulas. And certainly if this example is anything to go by, it begins to seem unlikely that the more complex kinds of patterns that we have seen so many times in this book could ever realistically be represented by such formulas.

But it turns out that there are at least some cases where traditional mathematical formulas can be found even though to the eye or with respect to other methods of perception and analysis a pattern may seem highly complex.

The picture at the top of the facing page is one example. A pattern is built up by superimposing a sequence of repetitive grids, and to the eye this pattern seems highly complex. But in fact there is a simple formula for the color of each square: given the largest factor in common between the



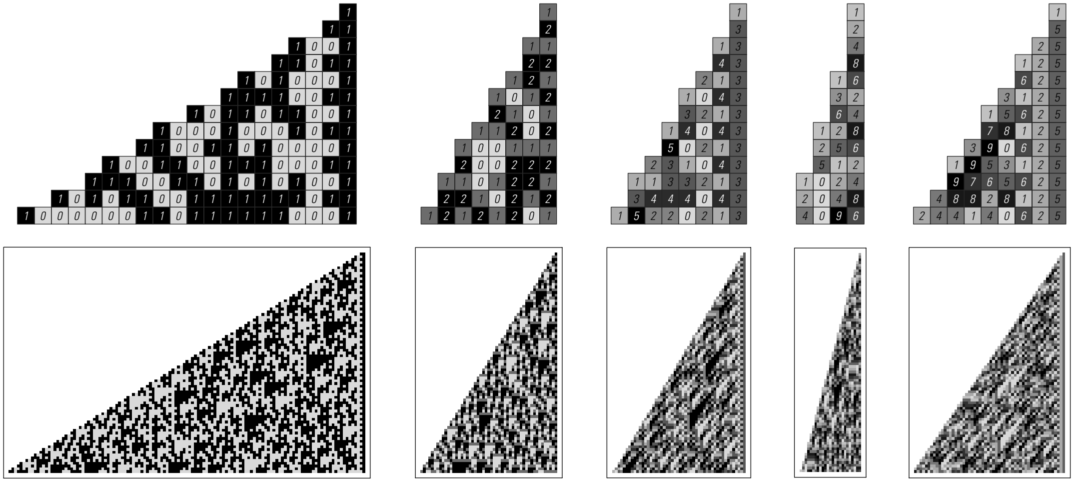
An example of a pattern that looks complex, but can nevertheless still be represented by a simple mathematical formula. Given the horizontal and vertical positions x and y a square is white when $GCD[x, y] = 1$ and is black otherwise. The condition $GCD[x, y] = 1$ is equivalent to the statement that x and y are relatively prime, or that no reduction is required to bring the fraction x/y to lowest terms. It can be shown that if the pattern is extended sufficiently far, then any possible local arrangement of black squares will eventually appear—though not necessarily with equal frequency.

numbers that specify the horizontal and vertical positions of the square, the square is white whenever this factor is 1, and is black otherwise.

So what about systems like cellular automata that have definite rules for evolution? Are there ever cases in which patterns generated by such systems seem complex to the eye but can in fact be described by simple mathematical formulas?

I know of one class of examples where this happens, illustrated in the pictures on the next page. The idea is to set up a row of cells corresponding to the digits of a number in a certain base, and then at each step to multiply this number by some fixed factor.

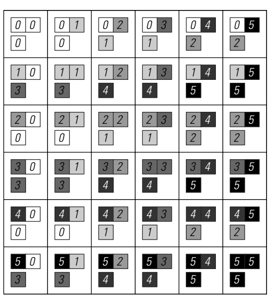
Such a system has many features immediately reminiscent of a cellular automaton. But at least in the case of multiplication by 3 in



base 2; multiplier 3 | base 3; multiplier 2 | base 6; multiplier 3 | base 10; multiplier 2 | base 10; multiplier 5

Patterns of digits in various bases generated by successive multiplication by a fixed factor. Such systems were discussed on page 120. With multiplier m row t corresponds to the power m^t. The value of the cell at position n from the end of row t is thus the n^{th} digit of m^t, or *Mod[Quotient[m^t, k^n], k]*. Despite the apparent complexity of the patterns, a fairly simple mathematical formula thus exists for the color of each square they contain.

base 2, the presence of carry digits in the multiplication process makes the system not quite an ordinary cellular automaton. It turns out, however, that multiplication by 3 in base 6, or by 2 or 5 in base 10, never leads to carry digits, with the result that in such cases the system can be thought of as following a purely local cellular automaton rule of the kind illustrated below.



	0	1	2	3	4	5
0	0	0	1	1	2	2
1	3	3	4	4	5	5
2	0	0	1	1	2	2
3	3	3	4	4	5	5
4	0	0	1	1	2	2
5	3	3	4	4	5	5

base 6; multiplier 3

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	1	1	1	1
1	2	2	2	2	2	3	3	3	3	3
2	4	4	4	4	4	5	5	5	5	5
3	6	6	6	6	6	7	7	7	7	7
4	8	8	8	8	8	9	9	9	9	9
5	0	0	0	0	0	1	1	1	1	1
6	2	2	2	2	2	3	3	3	3	3
7	4	4	4	4	4	5	5	5	5	5
8	6	6	6	6	6	7	7	7	7	7
9	8	8	8	8	8	9	9	9	9	9

base 10; multiplier 2

	0	1	2	3	4	5	6	7	8	9
0	0	0	1	1	2	2	3	3	4	4
1	5	5	6	6	7	7	8	8	9	9
2	0	0	1	1	2	2	3	3	4	4
3	5	5	6	6	7	7	8	8	9	9
4	0	0	1	1	2	2	3	3	4	4
5	5	5	6	6	7	7	8	8	9	9
6	0	0	1	1	2	2	3	3	4	4
7	5	5	6	6	7	7	8	8	9	9
8	0	0	1	1	2	2	3	3	4	4
9	5	5	6	6	7	7	8	8	9	9

base 10; multiplier 5

Cellular automaton rules equivalent to multiplication of digit sequences in various bases. The left part of the picture shows the explicit form of the rule for base 6 and multiplier 3. The arrays of numbers summarize the rule for this case and other cases. Note that only certain specific choices of base and multiplier lead to ordinary cellular automata; with other choices there are carries that propagate arbitrarily far. (See page 661.)

As the pictures at the top of the facing page demonstrate, the overall patterns produced in all cases tend to look complex, and in many respects random. But the crucial point is that because of the way the system was constructed there is nevertheless a simple formula for the color of each cell: it is given just by a particular digit in the number obtained by raising the multiplier to a power equal to the number of steps. So despite their apparent complexity, all the patterns on the facing page can in effect be described by simple traditional mathematical formulas.

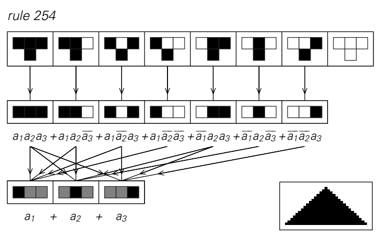
But if one thinks about actually using such formulas one might at first wonder what good they really are. For if one was to work out the value of a power m^t by explicitly performing t multiplications, this would be very similar to explicitly following t steps of cellular automaton evolution. But the point is that because of certain mathematical features of powers it turns out to be possible—as indicated in the table below—to find m^t with many fewer than t operations; indeed, one or two operations for every base 2 digit in t is always for example sufficient.

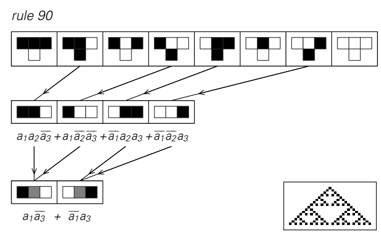
m^1	m	m
m^2	$m \times m$	m^2
m^3	$m \times m \times m$	$m^2 \times m$
m^4	$m \times m \times m \times m$	$(m^2)^2$
m^5	$m \times m \times m \times m \times m$	$(m^2)^2 \times m$
m^6	$m \times m \times m \times m \times m \times m$	$(m^2 \times m)^2$
m^7	$m \times m \times m \times m \times m \times m \times m$	$(m^2 \times m)^2 \times m$
m^8	$m \times m \times m \times m \times m \times m \times m \times m$	$((m^2)^2)^2$
m^9	$m \times m \times m \times m \times m \times m \times m \times m \times m$	$((m^2)^2)^2 \times m$
m^{10}	$m \times m \times m \times m \times m \times m \times m \times m \times m \times m$	$((m^2)^2 \times m)^2$

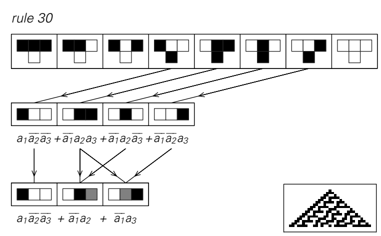
Examples of how powers can be computed more efficiently than by successive multiplications. In the cases shown, the choice of whether to square or multiply by an additional factor of m at each step in computing m^t is made on the basis of the successive digits in the base 2 representation of the number t.

So what about other patterns produced by cellular automata and similar systems? Is it possible that in the end all such patterns could just be described by simple mathematical formulas? I do not think so. In fact, as I will argue in Chapter 12, my strong belief is that in the vast majority of cases it will be impossible for quite fundamental reasons to find any such simple formula. But even though no simple formula may exist, it is still always in principle possible to represent the outcome of any process of cellular automaton evolution by at least some kind of formula.

The picture below shows how this can be done for a single step in the evolution of three elementary cellular automata. The basic idea is to translate the rule for a given cellular automaton into a formula that depends on three variables a_1, a_2 and a_3 whose values correspond to the colors of the three initial cells. The formula consists of a sum of terms, with each term being zero unless the colors of the three cells match a situation in which the rule yields a black cell.



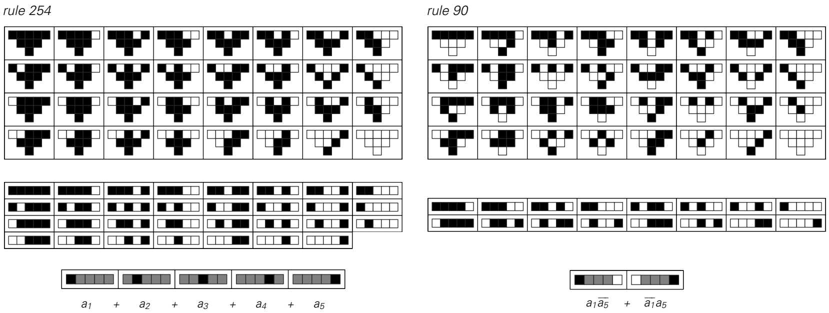


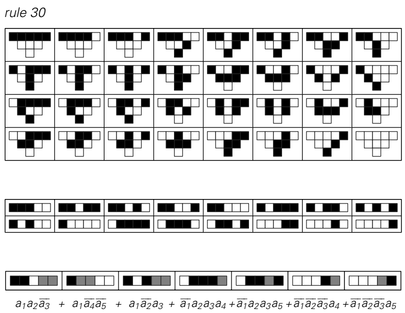


Boolean expression representations of the rules for three elementary cellular automata. The first row shows the original cellular automaton rules. The second row shows those combinations of cells that yield a black cell according to each of the rules. The third row shows a minimized version in which gray cells are introduced to indicate either black or white. In the formulas under the second and third rows the variable a_i represents the color of the i^{th} cell. $e_i \, e_j$ is analogous to $e_i \wedge e_j$, $e_i + e_j$ to $e_i \vee e_j$, and $\overline{e_i}$ to $\neg e_i$. The formulas given are in so-called disjunctive normal form (DNF). They are set up so that only at most one term in each formula is ever relevant for any particular configuration of colors.

In the first instance, each term can be set up to correspond directly to one of the cases in the original rule. But in general this will lead to a more complicated formula than is necessary. For as the picture demonstrates, it is often possible to combine several cases into one term by ignoring the values of some of the variables.

The picture at the top of the facing page shows what happens if one considers two steps of cellular automaton evolution. There are now altogether five variables, but at least for rules like rules 254 and 90 the individual terms end up not depending on most of these variables.





Boolean expression representations of the results from two steps in the evolution of three elementary cellular automata. At the top in each case is shown the explicit array of outcomes for each of the 32 possible initial configurations of cells. In the middle are shown those configurations that yield black cells. And at the bottom are the minimal representations of these collections of possibilities.

So what happens if one considers more steps? As the pictures on the next page demonstrate, rules like 254 and 90 that have fairly simple behavior lead to formulas that stay fairly simple. But for rule 30 the formulas rapidly get much more complicated.

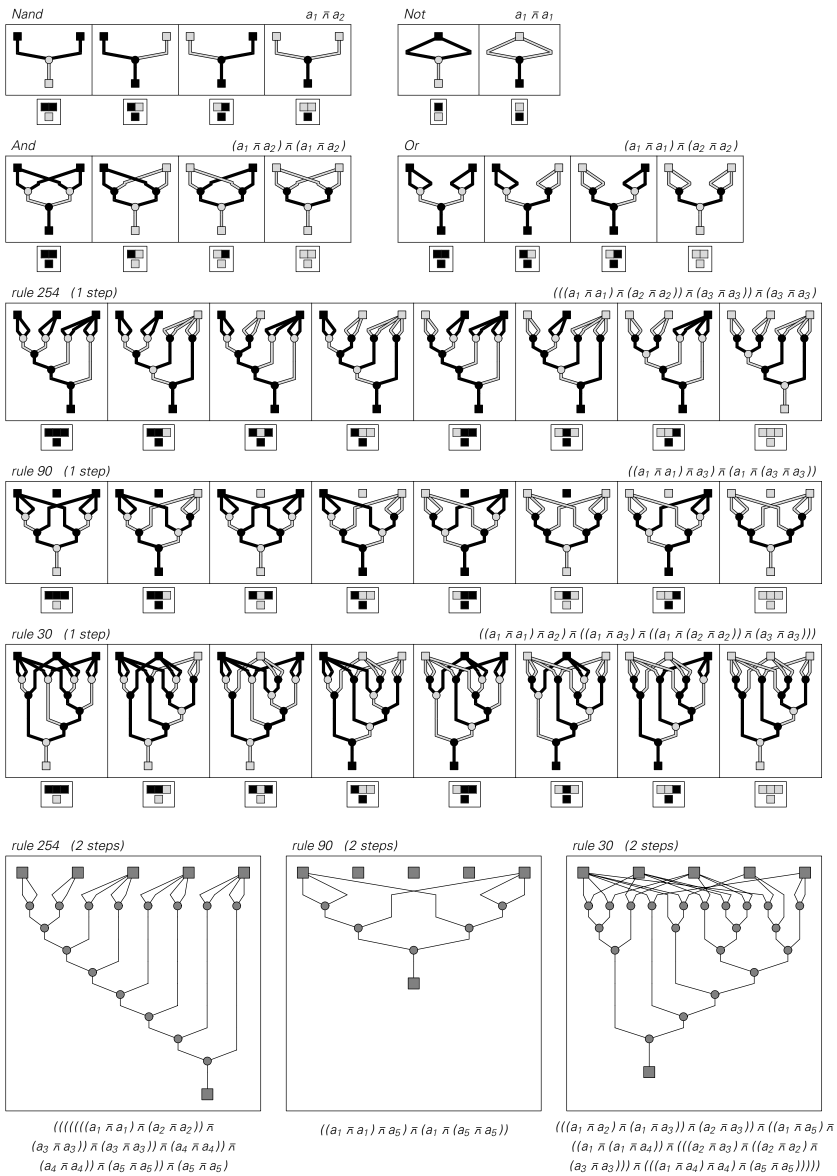
So this strongly suggests that no simple formula exists—at least of the type used here—that can describe patterns generated by any significant number of steps of evolution in a system like rule 30.

But what about formulas of other types? The formulas we have used so far can be thought of as always consisting of sums of products of variables. But what if we allow formulas with more general structure, not just two fixed levels of operations?

It turns out that any rule for blocks of black and white cells can be represented as some combination of just a single type of operation—for example a so-called NAND function of the kind often used in digital electronics. And given this, one can imagine finding for any particular rule the formula that involves the smallest number of NAND functions.

Minimal Boolean expression representations for the results of steps 1 through 5 in the evolution of three elementary cellular automata. Both rules 254 and 90 have fairly simple overall behavior, and yield comparatively small Boolean expressions. Rule 30 has much more complicated behavior and yields Boolean expressions whose size grows rapidly from one step to the next. (For steps 1 through 6, the expressions involve 3, 7, 17, 41, 102 and 261 terms respectively.) In each case the Boolean expressions given are the smallest possible in the disjunctive normal form (DNF) used.

The picture below shows some examples of the results. And once again what we see is that for rules with fairly simple behavior the formulas are usually fairly simple. But in cases like rule 30, the formulas one gets are already quite complicated even after just two steps.



Minimal representations in terms of NAND functions of the first two steps in the evolution of the same cellular automata as on the facing page. In each case, the network and formula shown are ones that involve the absolute minimum number of operations. Finding these effectively required searching through billions of possibilities. The picture at the top left shows the action of a single NAND function. The next three pictures show how the operations used in DNF formulas can be built up from NANDs.

So even if one allows rather general structure, the evidence is that in the end there is no way to set up any simple formula that will describe the outcome of evolution for a system like rule 30.

And even if one settles for complicated formulas, just finding the least complicated one in a particular case rapidly becomes extremely difficult. Indeed, for formulas of the type shown on page 618 the difficulty can already perhaps double at each step. And for the more general formulas shown on the previous page it may increase by a factor that is itself almost exponential at each step.

So what this means is that just like for every other method of analysis that we have considered, we have little choice but to conclude that traditional mathematics and mathematical formulas cannot in the end realistically be expected to tell us very much about patterns generated by systems like rule 30.

Human Thinking

When we are presented with new data one thing we can always do is just apply our general powers of human thinking to it. And certainly this allows us with rather modest effort to do quite well in handling all sorts of data that we choose to interact with in everyday life. But what about data generated by the kinds of systems that I have discussed in this book? How does general human thinking do with this?

There are definitely some limitations, since after all, if general human thinking could easily find simple descriptions of, for example, all the various pictures in this book, then we would never have considered any of them complex.

One might in the past have assumed that if a simple description existed of some piece of data, then with appropriate thinking and intelligence it would usually not be too difficult to find it. But what the results in this book establish is that in fact this is far from true. For in the course of this book we have seen a great many systems whose underlying rules are extremely simple, yet whose overall behavior is sufficiently complex that even by thinking quite hard we cannot recognize its simple origins.