EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

SECTION 11.8

The Rule 110 Cellular Automaton

for example purely repetitive patterns. But the crucial point is that as one looks at cellular automata with progressively greater computational capabilities, one will eventually pass the threshold of universality. And once past this threshold, the set of computations that can be performed will always be exactly the same.

One might assume that by using more and more sophisticated underlying rules, one would always be able to construct systems with ever greater computational capabilities. But the phenomenon of universality implies that this is not the case, and that as soon as one has passed the threshold of universality, nothing more can in a sense ever be gained.

In fact, once one has a system that is universal, its properties are remarkably independent of the details of its construction. For at least as far as the computations that it can perform are concerned, it does not matter how sophisticated the underlying rules for the system are, or even whether the system is a cellular automaton, a Turing machine, or something else. And as we shall see, this rather remarkable fact forms the basis for explaining many of the observations we made in Chapter 3, and indeed for developing much of the conceptual framework that is needed for the new kind of science in this book.

The Rule 110 Cellular Automaton

In previous sections I have shown that a wide variety of different kinds of systems can in principle be made to exhibit the phenomenon of universality. But how complicated do the underlying rules need to be in a specific case in order actually to achieve universality?

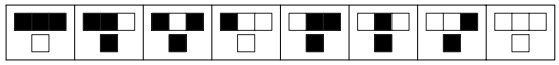
The universal cellular automaton that I described earlier in this chapter had rather complicated underlying rules, involving 19 possible colors for each cell, depending on next-nearest as well as nearest neighbors. But this cellular automaton was specifically constructed so as to make its operation easy to understand. And by not imposing this constraint, one might expect that one would be able to find universal cellular automata that have at least somewhat simpler underlying rules.

Fairly straightforward modifications to the universal cellular automaton shown earlier in this chapter allow one to reduce the number

of colors from 19 to 17. And in fact in the early 1970s, it was already known that cellular automata with 18 colors and nearest-neighbor rules could be universal. In the late 1980s—with some ingenuity—examples of universal cellular automata with 7 colors were also constructed.

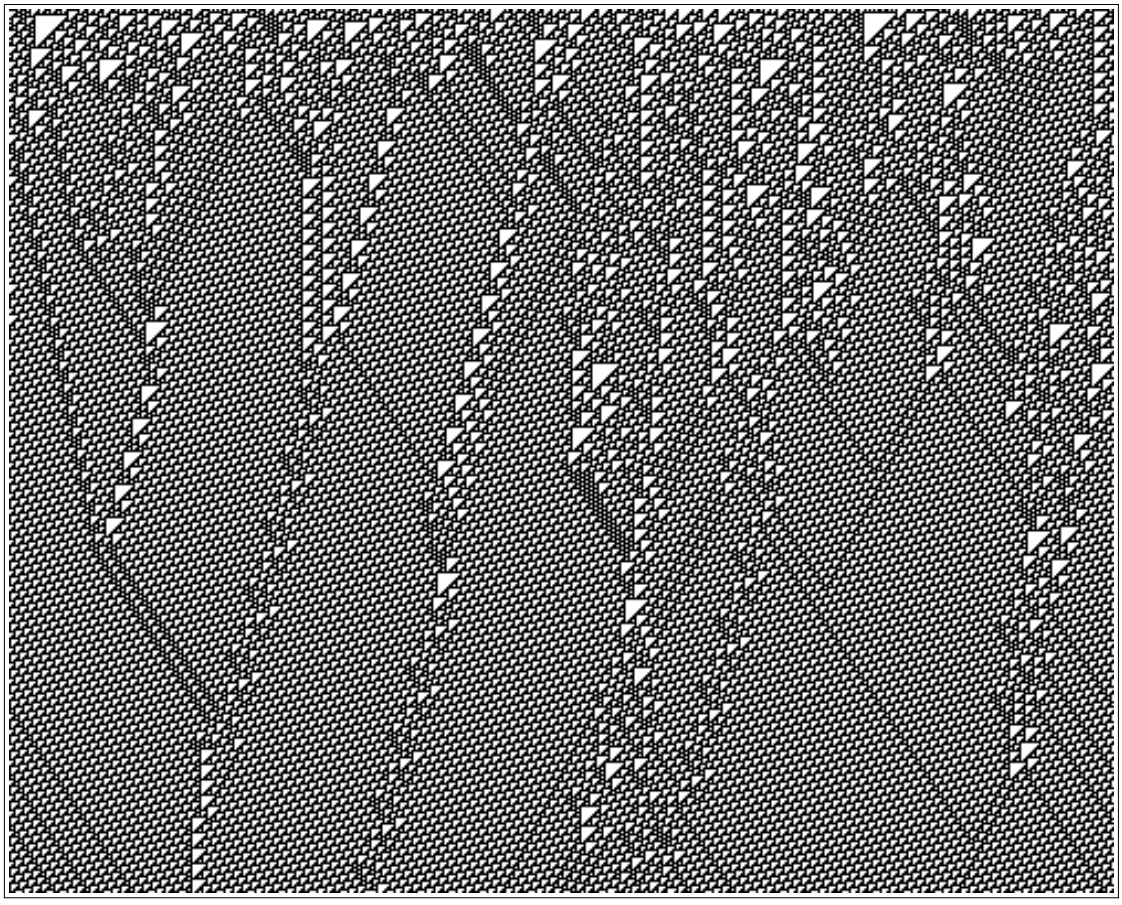
But such rules still involve 343 distinct cases and are by almost any measure very complicated. And certainly rules this complicated could not reasonably be expected to be common in the types of systems that we typically see in nature. Yet from my experiments on cellular automata in the early 1980s I became convinced that very much simpler rules should also show universality. And by the mid-1980s I began to suspect that even among the very simplest possible rules—with just two colors and nearest neighbors—there might be examples of universality.

The leading candidate was what I called rule 110—a cellular automaton that we have in fact discussed several times before in this book. Like any of the 256 so-called elementary rules, rule 110 can be specified as below by giving the outcome for each of the eight possible combinations of colors of a cell and its nearest neighbors.



The underlying rules for the rule 110 cellular automaton discussed in this section. As elsewhere in the book, each of the eight cases shows what the new color of a cell should be based on its own previous color, and on the previous colors of its neighbors. Despite the extreme simplicity of its underlying rules, what this section will demonstrate is that the rule 110 cellular automaton is in fact universal, and is thus in a sense capable of arbitrarily complex behavior. If the values of the cells in each block are labelled p, q and r, then rule 110 can be written as $Mod[(1 + p)\, q\, r + q + r, 2]$ or $\neg\, (p \wedge q \wedge r) \wedge (q \vee r)$.

Looking just at this very simple specification, however, it seems at first quite absurd to think that rule 110 might be universal. But as soon as one looks at a picture of how rule 110 actually behaves, the idea that it could be universal starts to seem much less absurd. For despite the simplicity of its underlying rules, rule 110 supports a whole variety of localized structures—that move around and interact in many complicated ways. And from pictures like the one on the facing page, it begins to seem not unreasonable that perhaps these localized structures could be arranged so as to perform meaningful computations.



A typical example of the behavior of rule 110 with random initial conditions. From looking at pictures like these one can begin to imagine that it could be possible to arrange localized structures in rule 110 so as to be able to perform meaningful computations. Note that page 292 already showed many of the types of localized structures that can occur in rule 110.

In the universal cellular automaton that we discussed earlier in this chapter, each of the various kinds of components involved in its operation had properties that were explicitly built into the underlying rules. Indeed, in most cases each different type of component was simply represented by a different color of cell. But in rule 110 there are only two possible colors for each cell. So one may wonder how one could ever expect to represent different kinds of components.

The crucial idea is to build up components from combinations of localized structures that the rule in a sense already produces. And if this works, then it is in effect a very economical solution. For it potentially allows one to get a large number of different kinds of components without ever needing to increase the complexity of the underlying rules at all.

But the problem with this approach is that it is typically very difficult to see how the various structures that happen to occur in a particular cellular automaton can be assembled into useful components.

And indeed in the case of rule 110 it took several years of work to develop the necessary ideas and tools. But finally it has turned out to be possible to show that the rule 110 cellular automaton is in fact universal.

It is truly remarkable that a system with such simple underlying rules should be able to perform what are in effect computations of arbitrary sophistication, but that is what its universality implies.

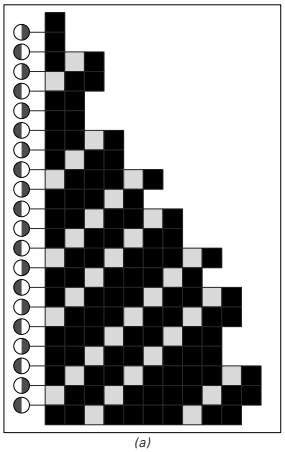
So how then does the proof of universality proceed?

The basic idea is to show that rule 110 can emulate any possible system in some class of systems where there is already known to be universality. And it turns out that a convenient such class of systems are the cyclic tag systems that we introduced on page 95.

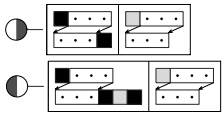
Earlier in this chapter we saw that it is possible to construct a cyclic tag system that can emulate any given Turing machine. And since we know that at least some Turing machines are universal, this fact then establishes that universal cyclic tag systems are possible.

So if we can succeed in demonstrating that rule 110 can emulate any cyclic tag system, then we will have managed to prove that rule 110 is itself universal. The sequence of pictures on the facing page shows the beginnings of what is needed. The basic idea is to start from the usual representation of a cyclic tag system, and then progressively to change this representation so as to get closer and closer to what can actually be emulated directly by rule 110.

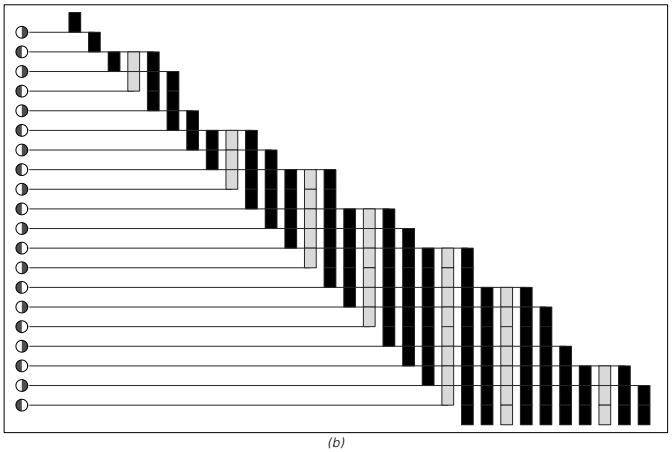
Picture (a) shows an example of the evolution of a cyclic tag system in the standard representation from pages 95 and 96. Picture (b) then shows another version of this same evolution, but now rearranged so that each element stays in the same position, rather than always shifting to the left at each step.



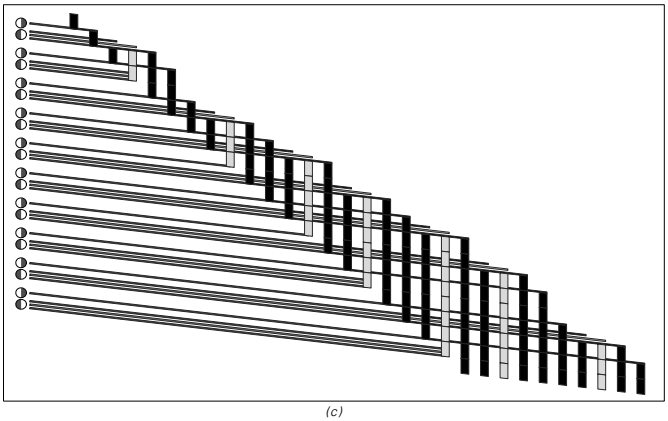
(a)



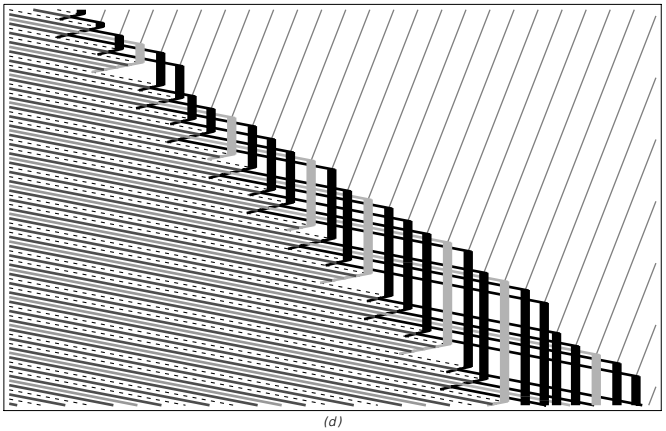
summary:



(b)



(c)



(d)

Four views of a cyclic tag system with rules as shown above, drawn so as to be progressively closer to what can be emulated directly in rule 110. Picture (a) shows the cyclic tag system in the same form as on pages 95 and 96. Picture (b) shows the system with sequences on successive steps rearranged so that they do not shift to the left when the first element is removed. Picture (c) is a skewed version of (b) in which the way information is used from the underlying rules at each step is explicitly indicated. Picture (d) shows a more definite mechanism for the evolution of the system in which different lines effectively indicate the motions of different pieces of information.

A cyclic tag system in general operates by removing the first element from the sequence that exists at each step, and then adding a new block of elements to the end of the sequence if this element is black. A crucial feature of cyclic tag systems is that the choice of what block of elements can be added does not depend in any way on the form of the sequence. So, for example, on the previous page, there are just two possibilities, and these possibilities alternate on successive steps.

Pictures (a) and (b) on the previous page illustrate the consequences of applying the rules for a cyclic tag system, but in a sense give no indication of an explicit mechanism by which these rules might be applied. In picture (c), however, we see the beginnings of such a mechanism.

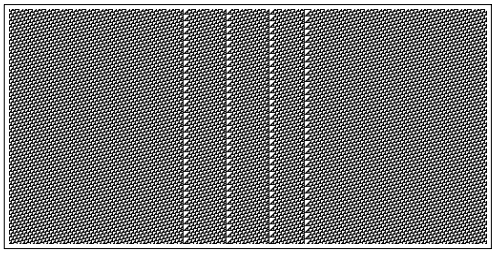
The basic idea is that at each step in the evolution of the system, there is a stripe that comes in from the left carrying information about the block that can be added at that step. Then when the stripe hits the first element in the sequence that exists at that step, it is allowed to pass only if the element is black. And once past, the stripe continues to the right, finally adding the block it represents to the end of the sequence.

But while picture (c) shows the effects of various lines carrying information around the system, it gives no indication of why the lines should behave in the way they do. Picture (d), however, shows a much more explicit mechanism. The collections of lines coming in from the left represent the blocks that can be added at successive steps. The beginning of each block is indicated by a dashed line, while the elements within the block are indicated by solid black and gray lines.

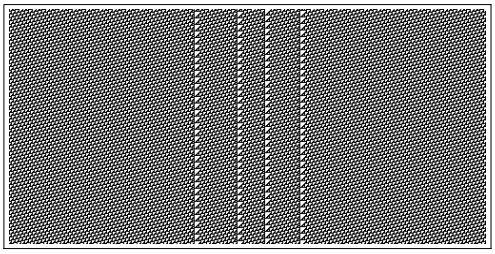
When a dashed line hits the first element in the sequence that exists at a particular step, it effectively bounces back in the form of a line propagating to the left that carries the color of the first element.

When this line is gray, it then absorbs all other lines coming from the left until the next dashed line arrives. But when the line is black, it lets lines coming from the left through. These lines then continue until they collide with gray lines coming from the right, at which point they generate a new element with the same color as their own.

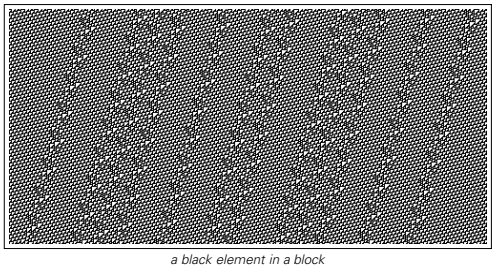
By looking at picture (d), one can begin to see how it might be possible for a cyclic tag system to be emulated by rule 110: the basic



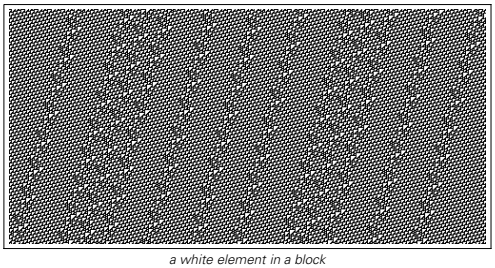
a black element in the sequence



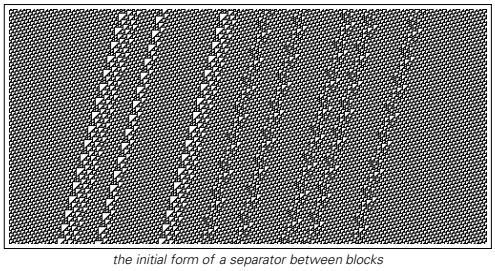
a white element in the sequence



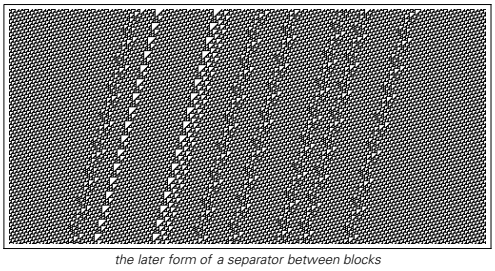
a black element in a block



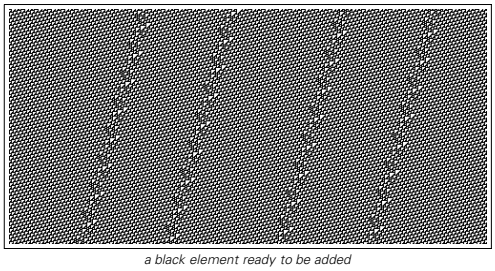
a white element in a block



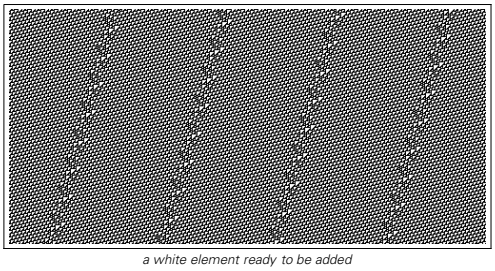
the initial form of a separator between blocks



the later form of a separator between blocks



a black element ready to be added



a white element ready to be added

Objects constructed from localized structures in rule 110, used for the emulation of cyclic tag systems. Each of the pictures shown is 500 cells wide. The objects in the top two pictures correspond to the thick vertical black and gray lines in picture (d) on page 679. The objects in the next two pictures correspond to the dark and light gray lines that come in from the left in picture (d). (Note that all the structures are left-right reversed in rule 110.) The third pair of pictures correspond to two versions of the dashed lines in picture (d). And the fourth pair of pictures correspond to right-going lines on the right-hand side of picture (d). All the localized structures involved in the pictures above were shown individually on page 292. Note that the spacings between structures are crucial in determining the objects they represent.

idea is to have each of the various kinds of lines in the picture be emulated by some collection of localized structures in rule 110.

But at the outset it is by no means clear that collections of localized structures can be found that will behave in appropriate ways.

With some effort, however, it turns out to be possible to find the necessary constructs, and indeed the previous page shows various objects formed from localized structures in rule 110 that can be used to emulate most of the types of lines in picture (d) on page 679.

The first two pictures show objects that correspond to the black and white elements indicated by thick vertical lines in picture (d). Both of these objects happen to consist of the same four localized structures, but the objects are distinguished by the spacings between these structures.

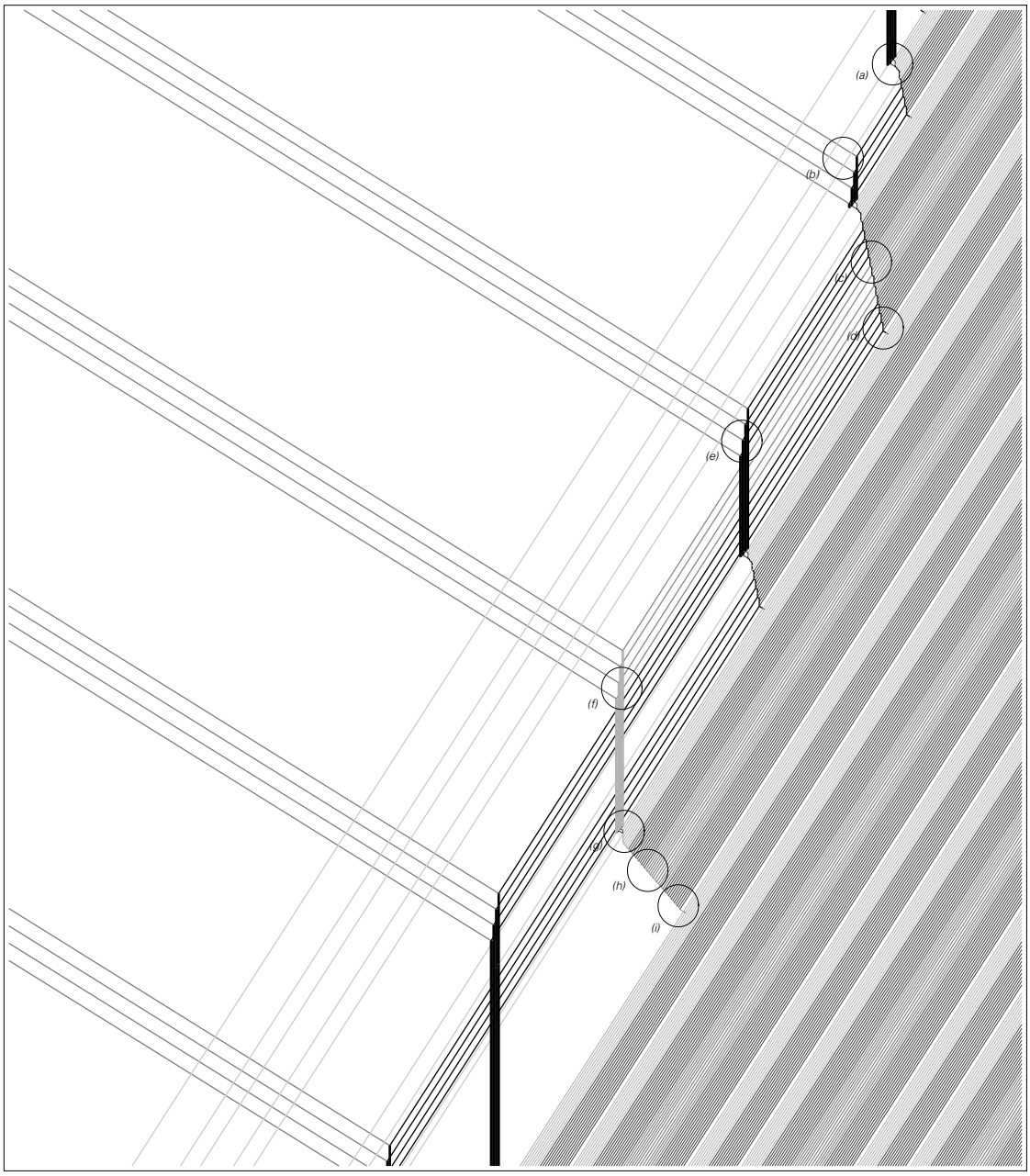
The second two pictures on the previous page use the same idea of different spacings between localized structures to represent the black and gray lines shown coming in from the left in picture (d) on page 679.

Note that because of the particular form of rule 110, the objects in the second two pictures on the previous page move to the left rather than to the right. And indeed in setting up a correspondence with rule 110, it is convenient to left-right reverse all pictures of cyclic tag systems. But using the various objects from the previous page, together with a few others, it is then possible to set up a complete emulation of a cyclic tag system using rule 110.

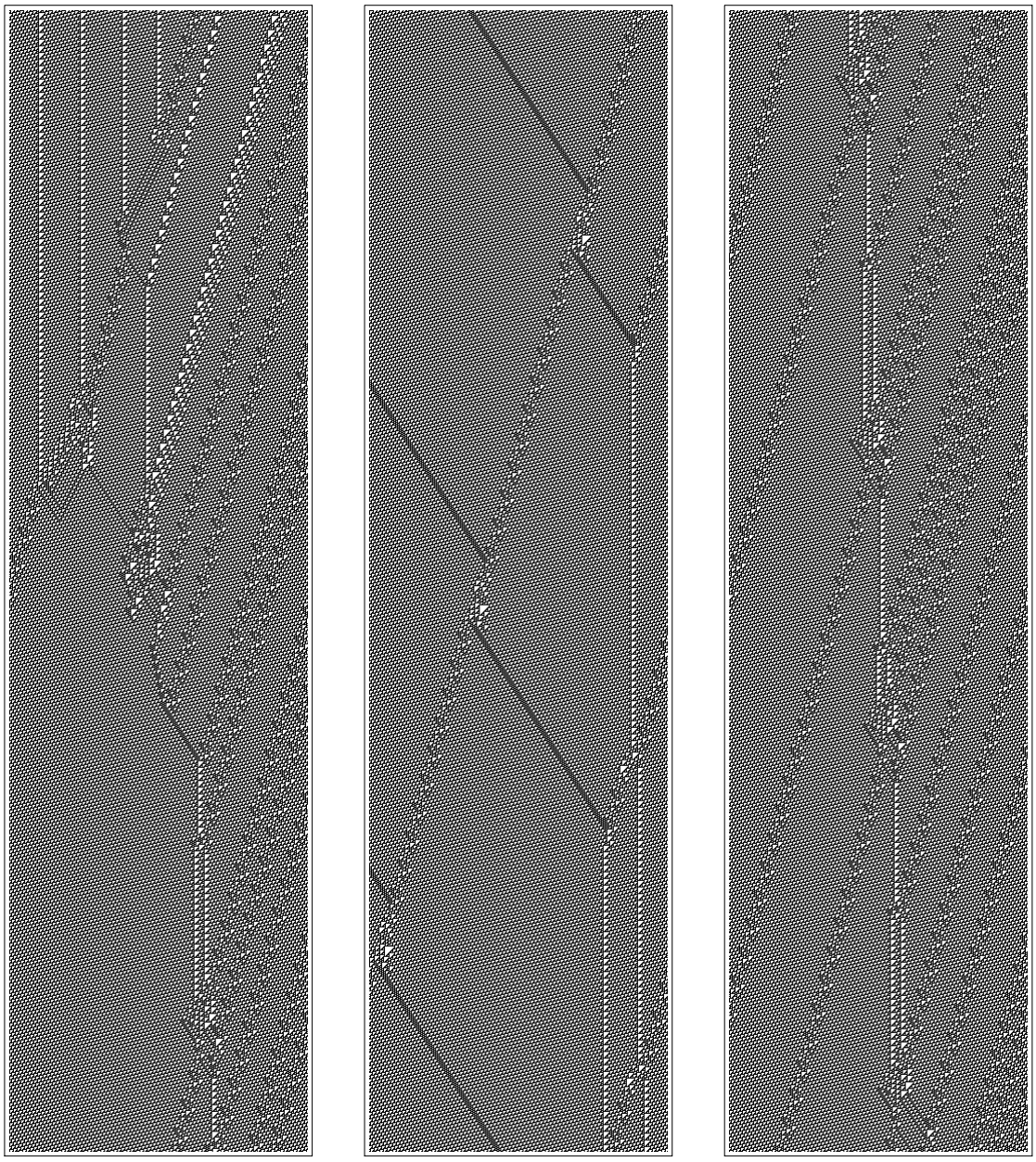
The diagram on the facing page shows schematically how this can be done. Every line in the diagram corresponds to a single localized structure in rule 110, and although the whole diagram cannot be drawn completely to scale, the collisions between lines correctly show all the basic interactions that occur between structures.

The next several pages then give details of what happens in each of the regions indicated by circles in the schematic diagram.

Region (a) shows a block separator—corresponding to a dashed line in picture (d) on page 679—hitting the single black element in the sequence that exists at the first step. Because the element hit is black, an object must be produced that allows information from the block at this step to pass through. Most of the activity in region (a) is concerned with producing such an object. But it turns out that as a side-effect two

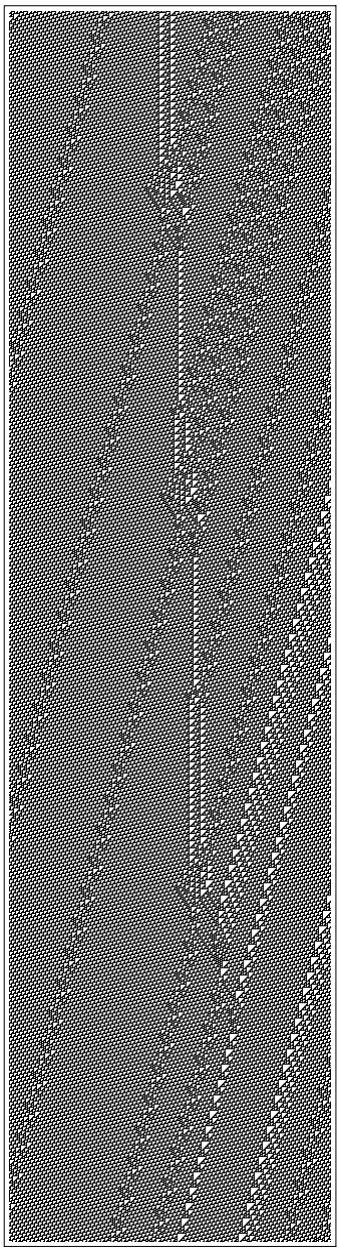


A schematic diagram of how rule 110 can be made to emulate a cyclic tag system. Each line in this diagram corresponds to one localized structure in rule 110. Note that the relative slopes of the structures are reproduced faithfully here, but their spacings are not. Note also that lines shown in different colors here often correspond to the same structure in rule 110.

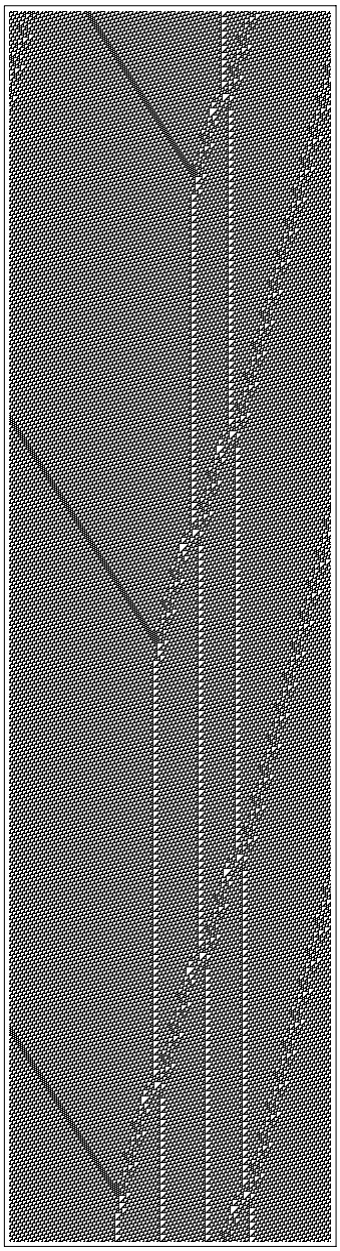


(a) (b) (c)

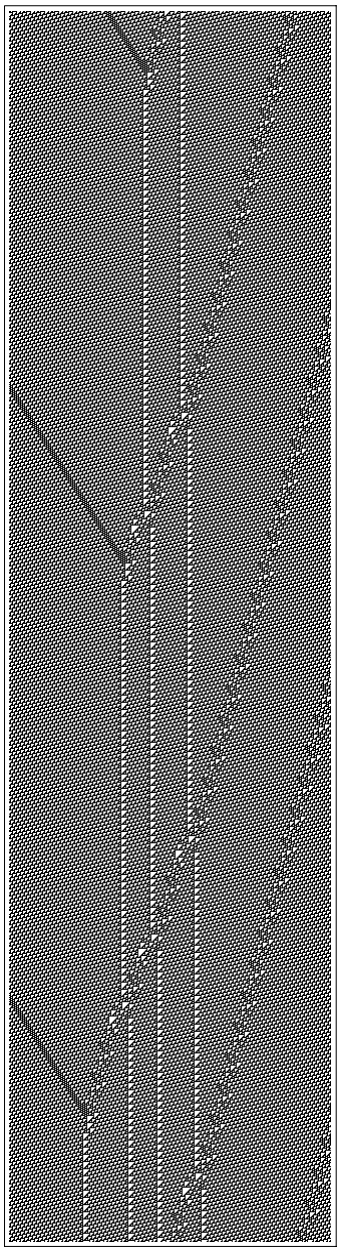
Close-ups of circled regions shown schematically on the previous page. Each picture is 320 cells wide and shows 1200 evolution steps.



(d)

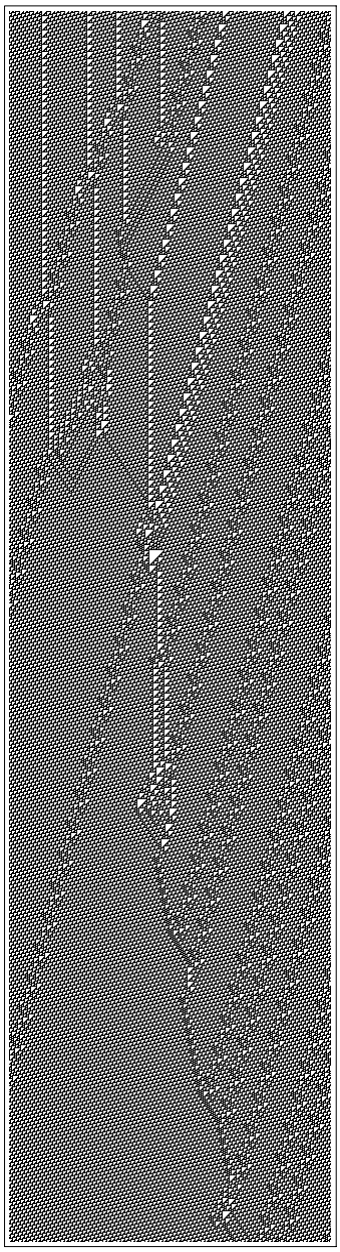


(e)

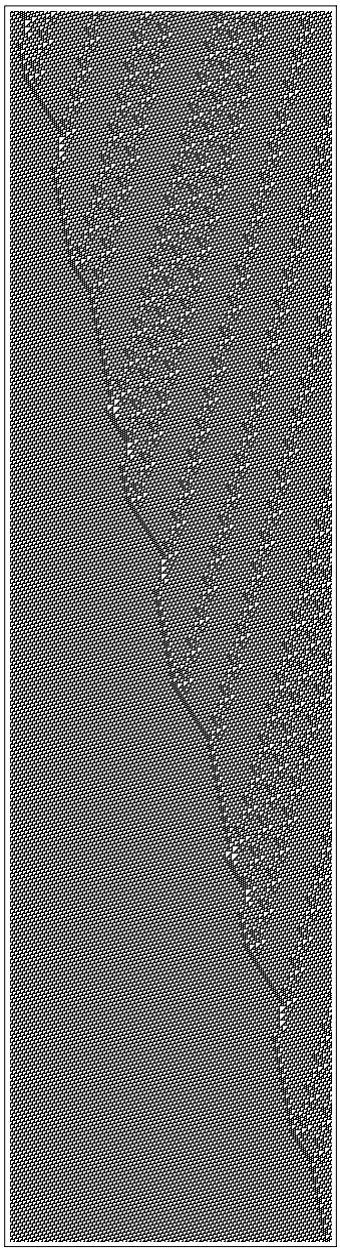


(f)

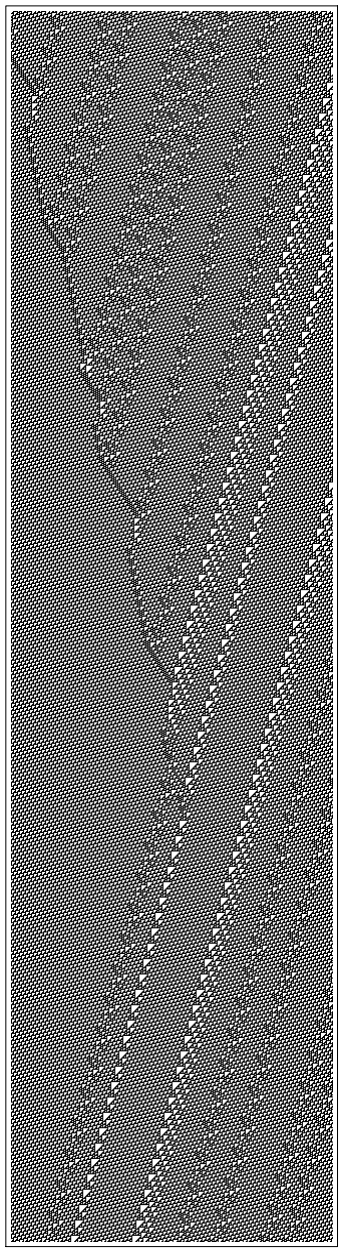
Close-ups (continued).



(g)



(h)



(i)

Close-ups (continued).

additional localized structures are produced that can be seen propagating to the left. These structures could later cause trouble, but looking at region (b) we see that in fact they just pass through other structures that they meet without any adverse effect.

Region (c) shows what happens when the information corresponding to one element in a block passes through the kind of object produced in region (a). The number of localized structures that represent the element is reduced from twelve to four, but the spacings of these structures continue to specify its color. Region (d) then shows how the object in region (c) comes to an end when the beginning of the block separator from the next step arrives.

Region (e) shows how the information corresponding to a black element in a block is actually converted to a new black element in the sequence produced by the cyclic tag system. What happens is that the four localized structures corresponding to the element in the block collide with four other localized structures travelling in the opposite direction, and the result is four stationary structures that correspond to the new element in the sequence.

Region (f) shows the same process as region (e) but for a white element. The fact that the element is white is encoded in the wider spacing of the structures coming from the right, which results in narrower spacing of the stationary structures.

Region (g) shows the analog of region (a), but now for a white element instead of a black one. The region begins much like region (a), except that the four localized structures at the top are more narrowly spaced. Starting around the middle of the region, however, the behavior becomes quite different from region (a): while region (a) yields an object that allows information to pass through, region (g) yields one that stops all information, as shown in regions (h) and (i).

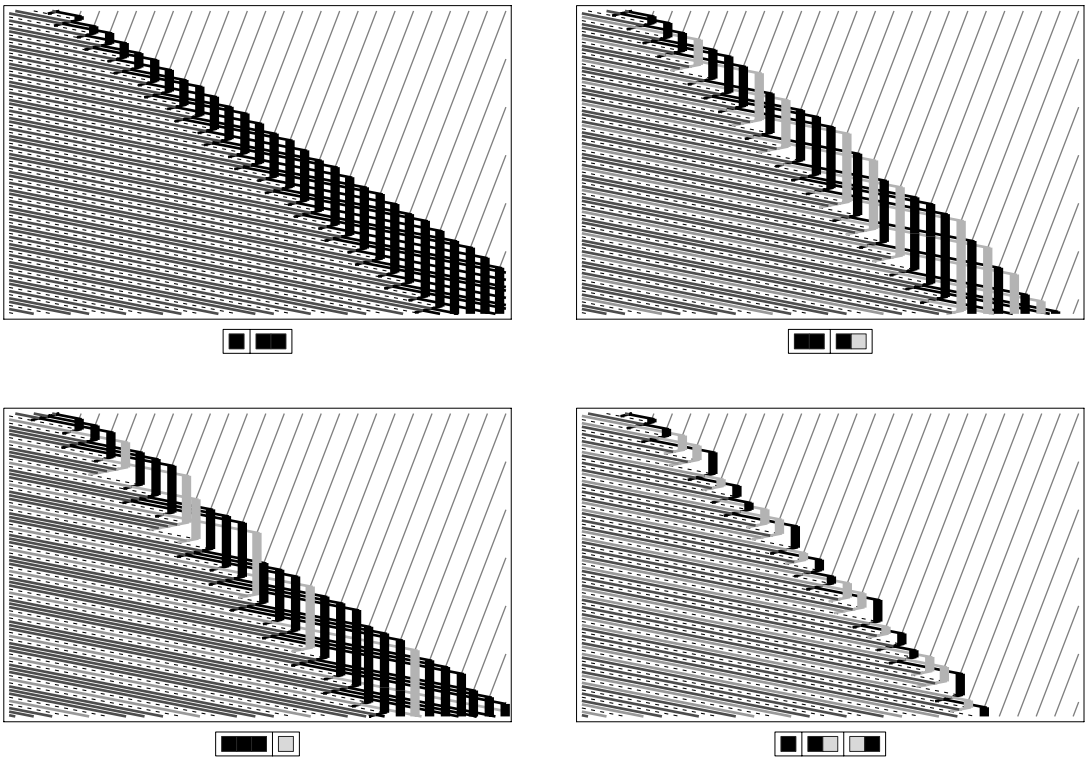
Note that even though they begin very differently, regions (d) and (i) end in the same way, reflecting the fact that in both cases the system is ready to handle a new block, whatever that block may be.

The pictures on the last few pages were all made for a cyclic tag system with a specific underlying rule. But exactly the same principles

can be used whatever the underlying rule is. And the pictures below show schematically what happens with a few other choices of rules.

The way that the lines interact in the interior of each picture is always exactly the same. But what changes when one goes from one rule to another is the arrangement of lines entering the picture.

In the way that the pictures are drawn below, the blocks that appear in each rule are encoded in the pattern of lines coming in from the left edge of the picture. But if each picture were extended sufficiently far to the left, then all these lines would eventually be seen to start from the top. And what this means is that the arrangement of lines can therefore always be viewed as an initial condition for the system.



Schematic diagrams of how cyclic tag systems with four different underlying rules can be emulated. The lines in each diagram correspond essentially to collections of localized structures in rule 110. The processes that occur in the interior of each picture are always the same; the different cyclic tag system rules are implemented by different arrangements of lines entering each picture.

This is then finally how universality is achieved in rule 110. The idea is just to set up initial conditions that correspond to the blocks that appear in the rule for whatever cyclic tag system one wants to emulate.

The necessary initial conditions consist of repetitions of blocks of cells, where each of these blocks contains a pattern of localized structures that corresponds to the block of elements that appear in the rule for the cyclic tag system. The blocks of cells are always quite complicated—for the cyclic tag system discussed in most of this section they are each more than 3000 cells wide—but the crucial point is that such blocks can be constructed for any cyclic tag system. And what this means is that with suitable initial conditions, rule 110 can in fact be made to emulate any cyclic tag system.

It should be mentioned at this point however that there are a few additional complications involved in setting up appropriate initial conditions to make rule 110 emulate many cyclic tag systems. For as the pictures earlier in this section demonstrate, the way we have made rule 110 emulate cyclic tag systems relies on many details of the interactions between localized structures in rule 110. And it turns out that to make sure that with the specific construction used the appropriate interactions continue to occur at every step, one must put some constraints on the cyclic tag systems being emulated.

In essence, these constraints end up being that the blocks that appear in the rule for the cyclic tag system must always be a multiple of six elements long, and that there must be some bound on the number of steps that can elapse between the addition of successive new elements to the cyclic tag system sequence.

Using the ideas discussed on page 669, it is not difficult, however, to make a cyclic tag system that satisfies these constraints, but that emulates any other cyclic tag system. And as a result, we may therefore conclude that rule 110 can in fact successfully emulate absolutely any cyclic tag system. And this means that rule 110 is indeed universal.