# Universality in Elementary Cellular Automata

**Matthew Cook**
*Department of Computation and Neural Systems,*[*]
*Caltech, Mail Stop 136-93,*
*Pasadena, California 91125, USA*

The purpose of this paper is to prove a conjecture made by Stephen Wolfram in 1985, that an elementary one dimensional cellular automaton known as "Rule 110" is capable of universal computation. I developed this proof of his conjecture while assisting Stephen Wolfram on research for *A New Kind of Science* [1].

## 1. Overview

The purpose of this paper is to prove that one of the simplest one dimensional cellular automata is computationally universal, implying that many questions concerning its behavior, such as whether a particular sequence of bits will occur, or whether the behavior will become periodic, are formally undecidable. The cellular automaton we will prove this for is known as "Rule 110" according to Wolfram's numbering scheme [2].

Being a one dimensional cellular automaton, it consists of an infinitely long row of cells $\{C_i \mid i \in \mathbb{Z}\}$. Each cell is in one of the two states $\{0, 1\}$, and at each discrete time step every cell synchronously updates itself according to the value of itself and its nearest neighbors: $\forall i, C_i' = F(C_{i-1}, C_i, C_{i+1})$, where $F$ is the following function:

$$
\begin{aligned}
F(0,0,0) &= 0\\
F(0,0,1) &= 1\\
F(0,1,0) &= 1\\
F(0,1,1) &= 1\\
F(1,0,0) &= 0\\
F(1,0,1) &= 1\\
F(1,1,0) &= 1\\
F(1,1,1) &= 0
\end{aligned}
$$

This $F$ encodes the idea that a cell in state 0 should change to state 1 exactly when the cell to its right is in state 1, and that a cell in state 1 should change to state 0 just when the cells on both sides are in state 1. Borrowing an analogy from the well known "Game of Life" two dimensional cellular automaton [3], we can think of a cell in state 1 as

[*]The work was performed when the author was with Wolfram Research, Inc.

a "live" cell, and a cell in state 0 as an "empty" cell. Then our function says that life spreads to the left, but live cells that have no room to breathe will die and become empty. For this reason, this automaton is also known as "LeftLife", a name which can help remind one how to mentally perform the updates.

If a cellular automaton has fewer than two states for the cells, or has two states but fewer than three cell values as arguments for $F$, then it is easy to show that universal behavior is impossible, so this result is the best possible in these regards.

There have been several previous results in the direction of finding small universal one dimensional cellular automata [4, 5, 6], but they have all been based on explicitly using the automaton's lookup table to implement the desired behavior, an approach which naturally requires a lookup table large enough to express the chosen universal algorithm.

The approach taken here is not to design a new cellular automaton, but to take the very simplest one that naturally exhibits complex behavior, and see if we can find within that complex behavior a way to make it do what we want. We will not concern ourselves directly with the lookup table given above, but instead we will look at the behavior that is naturally exhibited by the action of the automaton over time.

The automaton itself is so simple that its universality gives us a new tool for proving that other systems are universal. For example, we can construct Turing machines that are universal because they can emulate the behavior of Rule 110. These machines, shown in Figure 1, are far smaller than any previously known universal Turing machines [7].

## 2. Universal Systems

In the words of Minsky, a universal system is "a completely general instruction-obeying mechanism." When we say that some system is universal, or is capable of universal computation, we mean that it can run any program, or, in other words, execute any algorithm. Of course, the data for the program must be encoded in some form that the system can use, and the system's output must be similarly decoded. To make sure that the encoding and decoding processes aren't bearing the brunt of the computational burden of generating the output given the input, we typically require that they should be able to encode and decode the data fairly quickly, within some time limit that depends only on how much data needs to be converted.

The Church-Turing thesis [9] states that Turing machines[1] are capable of universal computation. In other words, given any algorithm, there is a way to have a Turing machine implement that algorithm, as long as

[1]We assume the reader has seen Turing machines before. If not, there is an introduction to them in [10].

| | 0 | 1 | $0^2$ | $1^2$ | $\neq$ |
|---|---|---|---|---|---|
| $S_E$ | $(0^2, S_E, L)$ | $(\neq, S_O, L)$ | $(0, S_E, R)$ | $(0, S_E, R)$ | $(1, S_O, R)$ |
| $S_O$ | $(\neq, S_E, L)$ | $(1^2, S_O, L)$ | $(0, S_E, R)$ | $(1, S_E, R)$ | $(1, S_O, R)$ |

| | $0_R$ | $1_R$ | $0_L$ | $1_L$ |
|---|---|---|---|---|
| $S_{x0}$ | $(0_L, S_{x0}, R)$ | $(1_L, S_{01}, R)$ | $(0_R, S_{x0}, L)$ | $(1_R, S_{x0}, L)$ |
| $S_{01}$ | $(1_L, S_{x0}, R)$ | $(1_L, S_{11}, R)$ | $X$ | $X$ |
| $S_{11}$ | $(1_L, S_{x0}, R)$ | $(0_L, S_{11}, R)$ | $X$ | $X$ |

| | 0 | 1 | $B$ |
|---|---|---|---|
| $S_{x0}$ | $(0, S_{x0}, R)$ | $(1, S_{01}, R)$ | $(0, S_B, L)$ |
| $S_{01}$ | $(1, S_{x0}, R)$ | $(1, S_{11}, R)$ | $X$ |
| $S_{11}$ | $(1, S_{x0}, R)$ | $(0, S_{11}, R)$ | $X$ |
| $S_B$ | $(0, S_B, L)$ | $(1, S_B, L)$ | $(0, S_{x0}, R)$ |

| | 0 | 1 |
|---|---|---|
| $S_{x0}$ | $(0, T_{x0}, R)$ | $(1, T_{01}, R)$ |
| $S_{01}$ | $(1, T_{x0}, R)$ | $(1, T_{11}, R)$ |
| $S_{11}$ | $(1, T_{x0}, R)$ | $(0, T_{11}, R)$ |
| $S_L$ | $(0, T_{x0}, L)$ | $(1, T_{x0}, L)$ |
| $T_{x0}$ | $(1, S_{x0}, R)$ | $(0, S_L, L)$ |
| $T_{01}$ | $(1, S_{01}, R)$ | $X$ |
| $T_{11}$ | $(1, S_{11}, R)$ | $X$ |

**Figure 1.** Some small Turing machines which are universal due to being able to emulate the behavior of Rule 110 by going back and forth over an ever wider stretch of tape, each time computing one more step of Rule 110's activity. The column headings show the symbols, and the row headings show the states. Entries marked "X" are not used by the machine.[8]

you are willing to have the input and output be in the form of characters on a Turing machine tape. If we like, we can consider this to be the definition of an algorithm.

Then, to show that some other specific class of machines is also capable of universal computation, all we need to show is that given any Turing machine program, there is a machine from the other class that can run an equivalent program. The standard way of doing this is to present a "compiler", which can take any Turing machine program and data and compile it into a program and data for the other kind of machine. The existence of such a compiler demonstrates that those machines can do anything a Turing machine can do, so in particular they must be able to execute any algorithm, and they are universal. This compiler, like the data encoder, is required to always finish within a predictable amount of time, without regard to what the program being compiled is designed to do.

## 2.1 Tag Systems

As our first example of another universal system, we will consider tag systems, which were originally considered by Post [11]. A tag system is a machine that operates on a finite tape, reading symbols off the front of the tape, and writing symbols onto the end of the tape. At every step, the machine removes two symbols, and based solely on the first one, decides what to append to the end of the tape, according to a fixed lookup table that defines the machine.[2]

For example, if the tape is *ACDABBE*, and the machine's lookup table lists the appendant *CCDD* for the symbol *A*, then the next step of the machine will be to delete *AC* from the front of the tape, and since the first one was *A*, it will append *CCDD* to the end of the tape, yielding *DABBECCDD* as the new tape. The step after that would read two more symbols and add the appendant for *D*. As we can see, half of the symbols are ignored. When an appendant is placed on the end of the tape, the parity of the length of the tape determines which symbols within the appendant will eventually be used and which will be ignored.

To prove that tag systems are universal, we will show that the behavior of any Turing machine can be emulated by a tag system. Since the class of Turing machines which use just two symbols is known to be universal, we will show how to emulate a Turing machine that uses just two symbols.[3] Following a proof of Cocke [12], we will also restrict ourselves to Turing machines operating on a tape on which only a finite number of cells have a symbol other than 0.[4] This will allow us to think of the tape as two binary numbers, one on each side of the head, with the less significant digits towards the head and the more significant digits farther from the head.

To emulate a Turing machine with $m$ states, we will use a tag system with $10m$ symbols:

$$\{L_k, L_{k0}, L_{k1}, R_k, R_{k0}, R_{k1}, R_{k*}, H_k, H_{k0}, H_{k1}\},$$
$$k \in \text{ states of the Turing machine}$$

If the cells on the tape to the left of the Turing machine's head form the number $T_L$, and the cells on the tape to the right of the Turing machine's head form the number $T_R$, then our tag system tape will represent these

[2]More generally, a tag system can remove a fixed number $p$ of symbols at each step, still only considering the first one.

[3]The proof we present is easily extended to Turing machines that use $p$ symbols, if we use a tag system that removes $p$ symbols at each step.

[4]The proof can be extended to accomodate periodic patterns on the tape by using a special symbol to mark the end of the tape, and when the Turing machine encounters this symbol, it "writes" another period onto the end of the tape as if the tape were a number in base $p^{period}$.

numbers in unary like this:

$$H_{k1}H_{k0}[L_{k1}L_{k0}]^{T_L}[R_{k1}R_{k0}]^{T_R}$$

As we proceed, we will think of the tag system as alternately using and ignoring symbols, so we may present a tape whose first symbol will be ignored. The price for this is that we must pay attention to the alignment of the system: whether the first symbol will be used or ignored.

Here, the bit in the cell that the Turing machine's head is at is represented by the alignment of the tag system: If the bit is 1, then the tag system is aligned so that the "$k1$" cells will be read and the "$k0$" cells will be ignored. If the bit is 0, then the tag system will read the "$k0$" cells and ignore the "$k1$" cells.

As the tag system processes the tape, it must emulate the Turing machine writing a bit at its current location, moving the head left or right, and changing to a new state.

To move the head right, we must multiply the number of $L$s by 2 and the number of $R$s by $\frac{1}{2}$, whereas to move the head left, we would have to double the $R$s and halve the $L$s. To write the bit we must add either 0 or 1 units to the unary representation of the part that got doubled. Changing to the new state simply consists of making sure the appended symbols are subscripted by the new state $k'$. So we can use the following lookup table entries, whose exact forms vary as shown according to the Turing machine's action upon reading bit $z$ in state $k$, to implement these processes in the tag system:

$$\begin{aligned}
H_{kz} &\rightsquigarrow [R_{k'*}R_{k'*}]^a[H_{k'}]^b H_{k'}[L_{k'}L_{k'}]^c \\
L_{kz} &\rightsquigarrow L_{k'}[L_{k'}L_{k'}L_{k'}]^d \\
R_{kz} &\rightsquigarrow R_{k'}[R_{k'}R_{k'}R_{k'}]^e \\
z &\in \{0,1\}
\end{aligned}$$

$a$ : include when $kz$ indicates a move to the left, writing a 1.
$b$ : include when $z$ is 0.
$c$ : include when $kz$ indicates a move to the right, writing a 1.
$d$ : include when $kz$ indicates a move to the right.
$e$ : include when $kz$ indicates a move to the left.

Then, when the tag system processes all the symbols originally on the tape, the tape will be transformed as follows:

$$H_{k1}H_{k0}[L_{k1}L_{k0}]^{T_L}[R_{k1}R_{k0}]^{T_R}$$

$$\Downarrow$$

$$[R_{k'*}R_{k'*}]^a[H_{k'}]^b H_{k'}[L_{k'}L_{k'}]^{(d?2:\frac{1}{2})T_L+(c?1:0)}[R_{k'}R_{k'}]^{(e?2:\frac{1}{2})T_R}$$

We use the notation $(condition?x : y)$ to denote one of two values, depending on a condition: It represents $x$ if the condition is present, and $y$ otherwise.

The tag system will still be aligned so that if it had read a 1, it will read the odd positions on the tape as shown, ignoring the even ones, while if it read a 0 it will read the even positions and ignore the odd ones. We see that in either case, it will definitely read the final $H_{k'}$ shown, as the optional preceding one was inserted if and only if it would be ignored. So, with the lookup table entries for $R_{k*}$ given by

$$R_{k*} \rightsquigarrow R_k R_k$$

the tape will, in another few (or perhaps zero) steps, be the following, with the alignment such that the first symbol will be read, the second ignored, and so on:

$$H_{k'}\,[L_{k'}L_{k'}]^{(d?2:\frac{1}{2})T_L+(c?1:0)}[R_{k'}R_{k'}]^{(e?2:\frac{1}{2})T_R+(a?1:0)}$$

So now, the number of units representing the side of the tape that the Turing machine is moving away from has been multiplied by two (representing shifting away from the Turing machine head) and the output bit has been written by adding one, or not, to that number. On the other hand, the units representing the other side of the tape, that the Turing machine is moving towards, have been halved. This is almost correct, but it might leave half a unit, depending on the value at the cell that the Turing machine is moving onto. This half unit should be truncated if present, and the value should somehow be read by the tag system.

As it turns out, this will happen automatically given the above state of the tag system's tape. If there are a whole number of units representing one side of the tape (an even number of $L$s or $R$s), then each unit will be read exactly once by the tag system. But if there is an extra half unit (a single $L$ or $R$), then that half unit will be ignored by the tag system, since the tag system is only reading the first position, third position, and so on. Although it won't be read, the half unit if present will disturb the alignment so that the tag system will ignore the last symbol shown above rather than reading it.

Now, if the lookup table entries for $H_k$, $L_k$, and $R_k$ are given by:

$$\begin{array}{lcl} H_k & \rightsquigarrow & H_{k1}H_{k0} \\ L_k & \rightsquigarrow & L_{k1}L_{k0} \\ R_k & \rightsquigarrow & R_{k1}R_{k0} \end{array}$$

then as the tag system continues processing the tape, it will be transformed:

$$H_{k'}[L_{k'}L_{k'}]^{(d?2:\frac{1}{2})T_L+(c?1:0)}[R_{k'}R_{k'}]^{(e?2:\frac{1}{2})T_R+(a?1:0)}$$

$$\Downarrow$$

$$H_{k'1}H_{k'0}[L_{k'1}L_{k'0}]^{T'_L}[R_{k'1}R_{k'0}]^{T'_R}$$

and the alignment of the tag system will be such that if the symbol read by the Turing machine was a 1 (i.e. if there was a half unit present to disturb the alignment), then the symbols with subscripts containing 1 will be read, but if a 0 was read by the Turing machine, then symbols with subscripts containing 0 will now be read.

This tape is in exactly the same format (including alignment) as when we started examining the behavior of the tag system, so at this point the tag system has emulated exactly one step of the Turing machine.

As time goes on, it will emulate more steps of the Turing machine, thus executing whatever algorithm the Turing machine would have executed. Therefore, tag systems are capable of universal computation.

## 2.2 Cyclic Tag Systems

For our next example of a universal system, we will consider cyclic tag systems, a new class of machines similar to tag systems, but simplified so as not to require a random access lookup table.

Like tag systems, cyclic tag systems work on a finite tape which is read from the front and appended to based on what is read. But instead of a table mapping characters to appendants, the machine is defined by just a plain list of appendants. At each step, the machine moves on to the next appendant in the list (cyclically going back to the beginning when the end of the list is reached), and considers whether or not to append it. The choice is controlled by what is read from the front of the tape. Unlike other systems, cyclic tag systems do not allow an arbitrary alphabet; they intrinsically require the two letter alphabet $\{Y, N\}$. At each step, the system reads exactly one character, and if that character is $Y$, then it appends the appendant under consideration, while an $N$ causes the appendant to be skipped.

We can easily show that cyclic tag systems are universal by showing that they can emulate tag systems. Given a tag system, we can construct a cyclic tag system that emulates it in the following manner: We will use a string of $N$s containing a single $Y$ to represent a symbol of the tag system. If we number the $k$ symbols of the tag system from 1 to $k$, then symbol number $i$ can be represented in the cyclic tag system by a string of length $k$ in which the $i^{th}$ character is $Y$ while the rest of the characters are $N$. The tape can be converted in this way, and so can the appendants in the lookup table. If the lookup table is also sorted so that the $k$ appendants for the $k$ symbols appear in order according to our numbering of the symbols, and we then add $k$ appendants of length zero, the resulting list of $2k$ appendants will define our cyclic tag system[5].

It is easy to see how this cyclic tag system emulates the original tag system. Each time it starts reading a sequence of $k$ characters corresponding to a symbol of the original tag system, it will alternately be either at the beginning of its list of $2k$ appendants or exactly halfway through the list. If it is at the beginning of the list when it starts reading the sequence of $k$ characters, then the single $Y$ will cause exactly one appendant to be appended, namely the one corresponding to the one that the tag system would have appended. If it is halfway through the list when it starts reading a sequence of $k$ characters, then since the second half of the list consists entirely of appendants of length zero, even the single $Y$ will not cause anything to be appended to the tape, and so the entire sequence of $k$ characters will have no effect at all. This corresponds to the tag system ignoring a symbol. Just as the tag system alternates between appending a sequence based on the symbol and ignoring the symbol, the cyclic tag system alternates between using the first half of the list to append an encoded sequence based on the encoded symbol and using the second half of the list to ignore the encoded symbol. In this way, a cyclic tag system can precisely emulate a tag system's behavior, and thus cyclic tag systems are universal machines.

## 2.3 Glider Systems

As our penultimate example of a universal system, we will consider glider systems, which are machines that idealize the notion of a system containing moving particles. We will consider one dimensional glider systems, which are machines that manipulate moving points on a line. Each moving point is called a glider. Like the alphabets of previous systems, a glider system has a finite set of types of gliders. The type of a glider specifies a fixed direction and speed for the glider. When two gliders meet, a lookup table specifies what the result of the collision should be. The result is a set of zero or more gliders which emanate from the point of collision.[6]

Glider systems bear a strong resemblance both to what is seen in experimental particle physics, and to what is seen in experimental cellular automaton simulations.

[5]More generally, if the tag system removes $p$ symbols at each step, then we add $(p-1)k$ appendants of length zero, resulting in a list of $pk$ appendants.

[6]To prevent the gliders from taking advantage of the continuity of the system and having an infinite number of collisions in a finite time, the results of collisions are specified only when the collisions occur at least some time or distance $\epsilon$ apart from each other. It is also often useful to have temporary gliders that "decay" by having a spontaneous collision a fixed amount of time after their creation. However, neither of these issues will be of concern in this section.

In general, glider systems are very flexible, and it is fairly easy to create one with gliders that do what you want. However, our interest in them here is just to show how one might implement a cyclic tag system using a glider system, so that we can then use the same approach for Rule 110.

Our approach will be to have the tape be represented by stationary gliders, which we will refer to as *tape data*. We will orient the tape so that the front is to the right, and the end is to the left.[7] An entry from the list of appendants, containing *table data* gliders to represent its *Y*s and *N*s, will glide in from the right and hit the first piece of tape data, which represents the first tape element. If the tape data represents an *N*, then the table data for the appendant will be wiped out. But if the first piece of tape data represents a *Y*, then the table data for the appendant will travel through the rest of the tape, to be appended onto the left end. The way we will append it is by having *ossifier* gliders waiting after the end of the tape, and when a symbol of the appendant hits an ossifier, it gets turned into new tape data, i.e. new stationary gliders representing a new symbol at the end of the tape.

The process of wiping out an appendant, or transmitting it, is not quite as simple as the last paragraph made it appear. To aid in these processes we will find it useful to have a preparatory glider that precedes the appendant, which we will call a *leader*. When the leader hits a tape data *N*, the result is a *rejector* glider that procedes to hit and eliminate each symbol of table data, until eventually it hits the next leader, which absorbs it. On the other hand, when the leader hits a tape data *Y*, the result is an *acceptor* glider that crosses over each symbol of table data, converting each into a symbol of *moving data*, until it too eventually hits the next leader, and is absorbed. Each symbol of moving data will cross the tape and then hit an ossifier, which will convert it into tape data.

Figure 2 shows a time history of such a glider system, with the initial state shown at the top, and time increasing down the picture, so a glider with zero velocity appears as a vertical line, while a rightward moving glider slants to the right as it goes down. The ossifiers, in the upper left area, are shown as having a non-zero velocity, which will turn out later to be useful, but for now it might cause some concern, since it means the tape data is spaced unevenly, and if there is an extended period of time during which no moving data emerges from the tape, then an ossifier could actually bump into a piece of tape data, which would not be good. Fortunately, if the cyclic tag system being emulated is constructed along the lines of Section 2.2 to emulate a tag system which in turn is constructed along the lines of Section 2.1, so that all the

[7]This may feel “backwards”, but this is the orientation we will use later with Rule 110, so it is best to get used to it now.

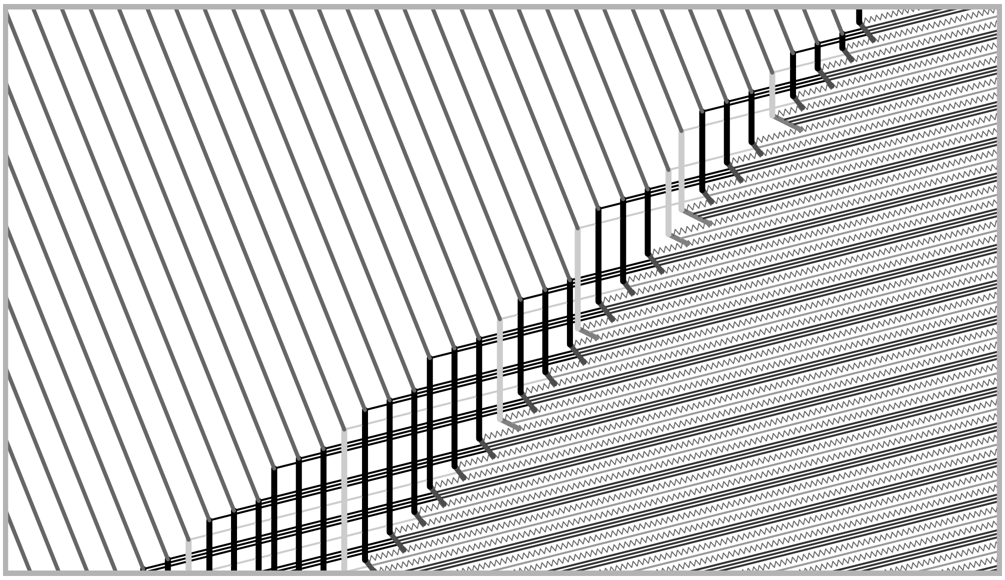

**Figure 2.** A glider system emulating a cyclic tag system which has a list of two appendants: *YYY* and *N*. Time starts at the top and increases down the picture. The gliders that appear to be entering on the sides actually start at the top, but the picture is not wide enough to show it. The gliders coming from the right are a periodic sequence, as are the ones on the left. The vertical stripes in the central chaotic swath are stationary gliders which represent the tape of the cyclic tag system, which starts here at the top with just a single *Y*. *Y*s are shown in black, and *N*s are shown in light gray. When a light gray *N* meets a leader (shown as a zig-zag) coming from the right, they produce a rejector which wipes out the table data until it is absorbed by the next leader. When a black *Y* meets a leader, an acceptor is produced, turning the table data into moving data which can cross the tape. After crossing the tape, each piece of moving data is turned into a new piece of stationary tape data by an ossifier coming from the left. Despite the simplicity of the appendant list and initial tape, this particular cyclic tag system appears to be immune to quantitative analysis, such as proving whether the two appendants are used equally often on average.

lookup table entries have a length greater than zero, then we know that at least once during the cycle of cyclic tag system appendants, we will append an appendant of positive length. This gives us an upper bound on the length of time between consecutive emerging pieces of moving data, which means that it is indeed possible to space the ossifiers in a fixed way so that they will always hit moving data and never tape data.

If the Turing machine that is ultimately being simulated is universal, then we can execute any program merely by encoding the program on its tape. This corresponds in the glider system to being able to encode any program in just the tape data portion of the initial arrangement, while always using a fixed repeating pattern on the right and on the left.

To find a specific example of an undecidable question in such a glider system, we can easily suppose that one of the Turing machine states represented in the tag system has the peculiar property that all of its $H$, $L$, and $R$ symbols lead to empty appendants for the end of the tape. Then, if this state is ever entered, the tag system will stop appending symbols to the tape, and the glider system will stop sending moving data to the end of the tape, causing ossifiers to hit tape data after all. This can lead to a new kind of glider being produced. So one undecidable question is, "Will the following glider ever appear?"

We could also easily suppose that one of the Turing machine states represented in the tag system has the property that it halves both the $L$ and $R$ symbols of the tape, and always stays in the same state. In this case, the tape will eventually be zeroed out, and the tag system's behavior will become periodic with period one, and the behavior of the glider system will likewise become periodic, with a specific predeterminable (*space*, *time*) period. So another example of an undecidable question is, "Will the behavior become periodic with the following period?"

If the glider system becomes periodic at all, then the emulated tag system must be periodic as well, meaning that the Turing machine is in an infinite loop where it keeps entering the same state, with the same tape to the left and right. Conversely, if the Turing machine enters such a loop, then the glider system must become periodic. Since it is undecidable whether a Turing machine will enter such a loop, another example of an undecidable question for the glider system is, "Will the behavior become periodic at all?"

In Section 4, we will show how Rule 110 can behave just like this kind of glider system, and the above undecidable questions will apply to it as well.

## 3. The Gliders of Rule 110

In this section, we will discuss the gliders of Rule 110, presenting concepts and tools that will be useful for the construction in Section 4.

Since we humans find it easier to identify objects in a two dimensional picture than a one dimensional picture, we will use space-time diagrams to show how Rule 110 acts on a row of cells. We will show cells in state 1 as black pixels, and cells in state 0 as white pixels, and successive rows of pixels in the picture will correspond to successive time steps. For example, in Figure 3, the top row of pixels is completely random. After each cell updates itself according to Rule 110's update function, the result is shown as the second row of pixels. Figure 3 shows the activity of 400 cells over a period of 250 time steps.

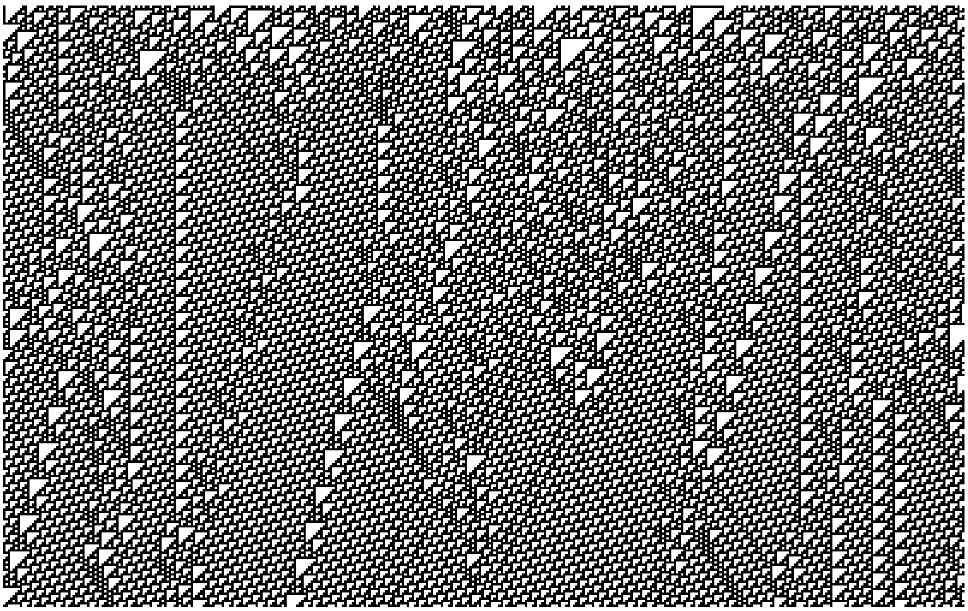

**Figure 3.** A space-time history of the activity of Rule 110, started at the top with a row of randomly set cells.

Although there are no known methods to prove that it should happen, we can plainly see that as time goes on, the cells stop being completely random, and in the lower part of the picture we can visually identify many phenomena a dozen or fewer cells wide that are periodic in time, and “move” through a lattice “background” of little white triangles. These periodic phenomena are the gliders, and the background lattice of white triangles is called *ether*.

Figure 4 shows all of the known gliders in Rule 110. One might think that the key to building constructions would be to find more gliders. However, when working with gliders in one dimension, it is useful to stick with ones that occur naturally. Indeed, every time two pieces of data cross each other in the confines of one dimension, they disrupt each other and must be “recreated”, as if by chance, after they collide. For this reason, it is much more profitable to always use gliders that are easily created in chance reactions than to try to invent or discover new gliders or other objects with special properties. Even using common gliders, it can be quite hard to find reactions that produce the desired gliders in the desired positions. Imagine how much harder it would be if the reaction had to produce some unnatural object!

Similarly, although Rule 110 can support other backgrounds besides the standard ether, such other backgrounds do not arise naturally, and in particular they would be extremely unlikely to re-arise in between the

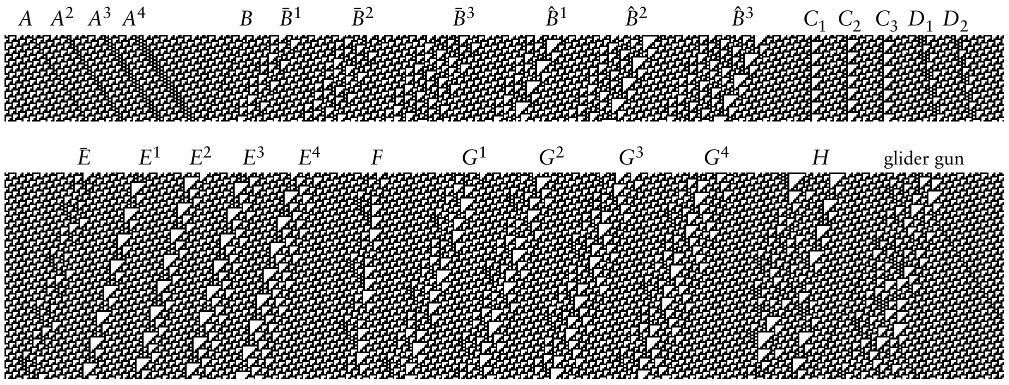
$A$ $A^2$ $A^3$ $A^4$ $B$ $\bar{B}^1$ $\bar{B}^2$ $\bar{B}^3$ $\hat{B}^1$ $\hat{B}^2$ $\hat{B}^3$ $C_1$ $C_2$ $C_3$ $D_1$ $D_2$
$\bar{E}$ $E^1$ $E^2$ $E^3$ $E^4$ $F$ $G^1$ $G^2$ $G^3$ $G^4$ $H$ glider gun

**Figure 4.** This shows all the known gliders that exist in the standard background, or ether, of Rule 110. Also, a "glider gun" is shown, which emits $A$ and $B$ gliders once per cycle. The lower gliders are shown for a longer time to make their longer periods more evident. $A$ gliders can pack very closely together, and $n$ such closely packed $A$s are denoted by $A^n$ as if they were a single glider. The other gliders with exponents are internally extendable, and the exponent can again be any positive integer, indicating how extended it is. The subscripts for $C$ and $D$ gliders indicate different alignments of the ether to the left of the glider, and may only have the values shown. Gliders are named by the same letter iff they have the same slope. The glider gun, $H$, $\hat{B}^n$, and $\bar{B}^{n\geq 2}$ are all rare enough that we say they do not arise naturally. Since the $\bar{B}^n$ arises naturally only for $n = 1$, $\bar{B}^1$ is usually written as just $\bar{B}$.

gliders produced by a reaction. In short, if we hope to create a pattern of coherent interaction among discernable entities, our best odds are if we take what Rule 110 willingly offers us, and play with it until we see how to build something out of it.

## 3.1 Glider Properties

One of the first things we can calculate for the gliders is their width, as given in Figure 5. Given a glider, if we consider the ether to its left compared with the ether to its right, we will probably find that these ether regions would not match up exactly if we were to extend them so that they overlapped. The ether to its right can be thought of as shifted $w$ cells to the right from where it would be if it were just a continuation of the ether on the left. Since the ether has a horizontal period of 14, the value of $w$ is a number mod 14, and we say that it is the *width* of the glider. We see that the sum of the widths of several adjacent gliders, mod 14, gives the offset of the ether on the far right compared to the ether on the far left, and so it is a conserved quantity that is not affected by collisions among the gliders.

| | | period | |
|---|---|---|---|
| | width | $(t, x)$ | $(\vec{A}, \vec{B})$ |
| $A$ | 6 | $(3, 2)$ | $(1, 0)$ |
| $B$ | 8 | $(4, -2)$ | $(0, 1)$ |
| $\bar{B}^n$ | $13 + 9n$ | $(12, -6)$ | $(0, 3)$ |
| $\hat{B}^n$ | $2 + 9n$ | $(12, -6)$ | $(0, 3)$ |
| $C_1$ | 9 | $(7, 0)$ | $(1, 1)$ |
| $C_2$ | 3 | $(7, 0)$ | $(1, 1)$ |
| $C_3$ | 11 | $(7, 0)$ | $(1, 1)$ |

| | | period | |
|---|---|---|---|
| | width | $(t, x)$ | $(\vec{A}, \vec{B})$ |
| $D_1$ | 11 | $(10, 2)$ | $(2, 1)$ |
| $D_2$ | 5 | $(10, 2)$ | $(2, 1)$ |
| $E^n$ | $11 + 8n$ | $(15, -4)$ | $(1, 3)$ |
| $\bar{E}$ | 7 | $(30, -8)$ | $(2, 6)$ |
| $F$ | 1 | $(36, -4)$ | $(4, 6)$ |
| $G^n$ | $2 + 8n$ | $(42, -14)$ | $(2, 9)$ |
| $H$ | 11 | $(92, -18)$ | $(8, 17)$ |

**Figure 5.** The width of a glider is the horizontal offset (mod 14) of the ether on its right with respect to the ether on its left. The period of each glider is given as a (time, horizontal displacement) pair. Due to the lattice structure of the background, this period can always also be expressed as a positive integral linear combination of the $A$ period and the $B$ period.

Since the sum of the widths is conserved mod 14, it is also conserved mod 2, and so the number of odd width gliders ($C$s, $D$s, $E$s, and $F$s are the only natural ones) is conserved mod 2.

Although more gliders are bound to exist, we can find some general limits on what speeds they might have. A case analysis of possible pixel values shows that the $A$ and various $B$ gliders travel at the maximum possible right and left speeds within the standard ether. The known $B$ forms are the only possible gliders that can travel at their speed, and the $A$ is the only possible glider that can travel to the right faster than a $D$.

If we think of a glider as including a little bit of the ether on each side, we see that its possible locations are constrained by the periodicity of the ether. If we want to move a glider, without changing the ether it lies in, we must move it by a vector which represents a periodicity of the background ether. If we call the period of an $A$ glider an $\vec{A}$ unit, and the period of a $B$ glider a $\vec{B}$ unit, then every periodicity of the ether is an integral combination of $\vec{A}$ units and $\vec{B}$ units, and conversely. Since a glider can be moved by its own period without changing anything, the period of each glider must be an integral combination of $\vec{A}$ and $\vec{B}$ units. Figure 5 shows how each glider's period can be expressed in this way.

It is an interesting exercise to show that the number of different kinds of collisions between two gliders is given by the "cross product" of their $(\vec{A}, \vec{B})$ periods: If one glider has an $(\vec{A}, \vec{B})$ period of $(p_a, p_b)$, and the other's is $(q_a, q_b)$, then the number of ways they can collide is $|p_a q_b - p_b q_a|$. As a simple example, the $A$ and $B$ gliders can collide in only one way, which results in mutual annihilation (note that their widths sum to 0 mod 14).

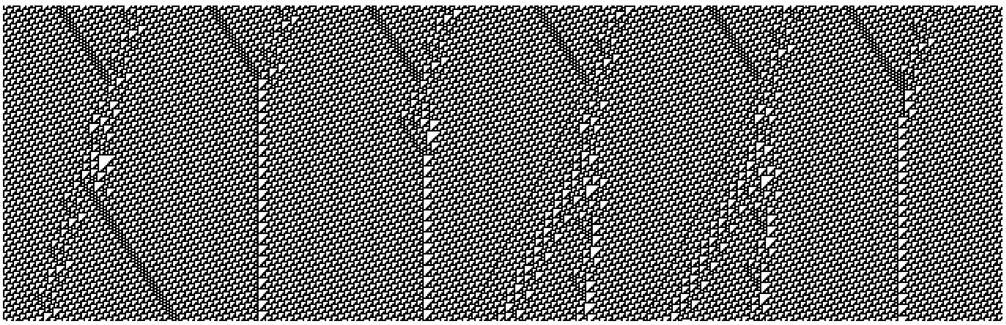

**Figure 6.** The six possible collisions between an $A^4$ and an $\bar{E}$.

## 3.2 Glider Measurements

Often we will want to compare the relative locations of nearby gliders. In fact, this is the only kind of measurement that will turn out to be useful. Absolute measurements, in terms of a fixed $(x, y)$ coordinate system, turn out to be almost impossible to work with, and so we won't use them.

For example, let's look at the six ways an $A^4$ can hit an $\bar{E}$, shown in Figure 6. The relative positions of the gliders as they collide determines which outcome will occur. What we need is some simple way to measure their relative positions. The way we will do this is with $\nearrow$ distance.

### 3.2.1 $\nearrow$ Distance

The idea of $\nearrow$ (pronounced "up") distance is to associate a diagonal row of ether triangles, parallel to the $A$ slope, with each glider. Since the ethers on the two sides of a glider are separated by the glider, we can do this independently on the two sides of the glider. For example, for an $\bar{E}$, we associate the rows shown in Figure 7. Then, if we want to compare two $\bar{E}$s which have no other gliders in between them, such as the two shown in Figure 7, we look in the ether region between them to see how the second $\bar{E}$'s rows compare to the first $\bar{E}$'s rows. If their rows match, then we say the second $\bar{E}$ is $\nearrow_0$ from the first. If a row of the second $\bar{E}$ is one row to the upper right from a row of the first $\bar{E}$, then we say the second $\bar{E}$ is $\nearrow_1$ from the first, and so on. Since $\bar{E}$s associate themselves with every sixth row due to their periodicity, the $\nearrow$ distance between them is necessarily a value mod 6. In Figure 7, the second $\bar{E}$ is $\nearrow_5$ from the first.

For an $A^n$ glider, the associated ether row on each side is simply the row adjacent to the glider. Now we know how to describe the six relative positions of the $A^4$ and $\bar{E}$ in Figure 6: They are the cases where the $\bar{E}$ is $\nearrow_0$, $\nearrow_1$, $\nearrow_2$, $\nearrow_3$, $\nearrow_4$, and $\nearrow_5$ from the $A^4$.

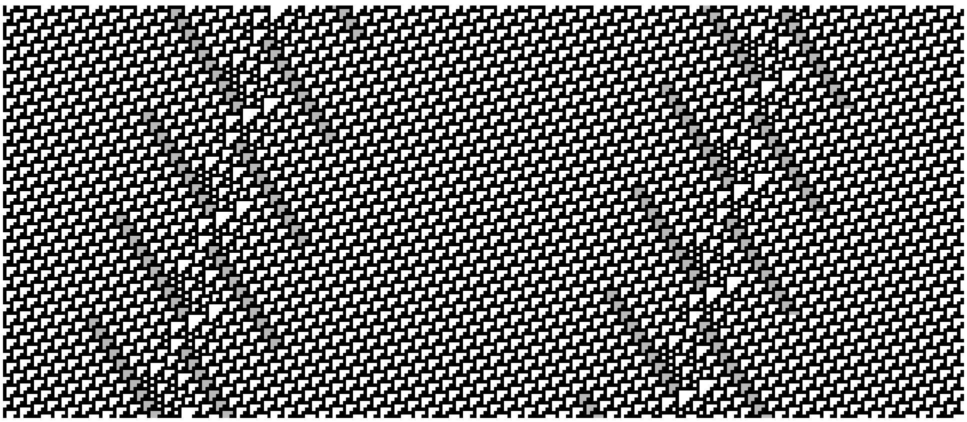

**Figure 7.** The $\nearrow$ distance for $\bar{E}$s is defined by associating diagonal rows of ether triangles with the $\bar{E}$s as shown. On each side of an $\bar{E}$, we associate it with the rows that penetrate farthest into the $\bar{E}$.

In this paper, our concern with $\nearrow$ distance will almost always ultimately be in relation to $\bar{E}$s, so for the remainder of the paper, it should be understood that all $\nearrow$ distances are a value mod 6. The only exception will be that when we are comparing two $A$ gliders, we might be interested in the absolute distance between them, in which case we will write it in small type between the $A$s, so for example $A\,{\scriptstyle 0}\,A$ represents two $A$s with just one diagonal row of ether triangles between them, so that both $A$s associate themselves with this row. By extension, $A\,{\scriptstyle -1}\,A$ is the same as $A^2$.

### 3.2.2 $\frown$ Distance

The idea of $\frown$ (pronounced "over") distance is to associate a vertical column of ether triangles with each glider. Since the ether triangles in a vertical column are not adjacent to each other, such a column does not make itself as clear to our eyes as did the diagonal rows used for $\nearrow$ distance. Nonetheless, it is fairly easy to measure the distance between such columns. If we move one column over, the triangles of the new column are staggered with respect to the original column. If we move two columns over, then the triangles are at about the same height and adjacent to the original triangles.[8]

For an $\bar{E}$, we associate the columns shown in Figure 8. Since $\bar{E}$s repeat themselves every fourth column, the $\frown$ distance to or from an $\bar{E}$

[8]Since $\frown$ distance measures pure horizontal displacement, we could in this case make equivalent measurements by using absolute space time coordinates and considering just the spatial coordinate. However, the use of $\frown$ distance better illuminates the general method of using the background's periodicity as an aid to practical measurement.

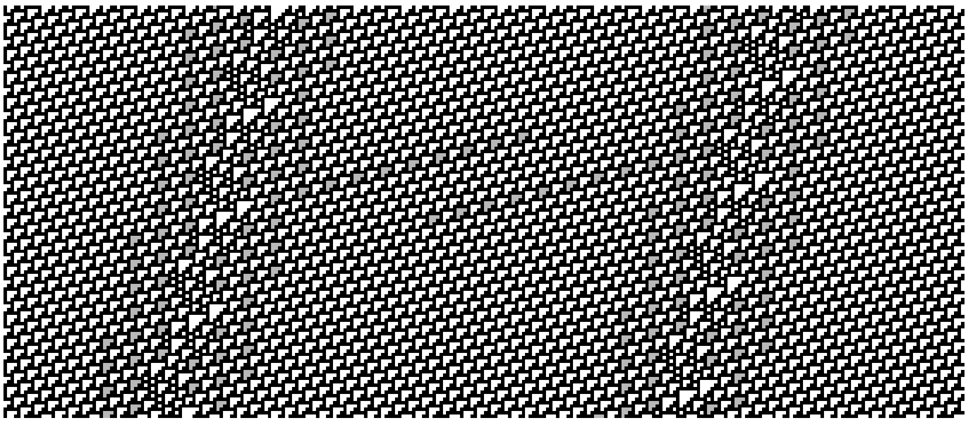

**Figure 8.** The $\frown$ distance for $\bar{E}$s is defined by associating vertical columns of ether triangles with each $\bar{E}$ as shown. The markings extending to the middle of the picture mark every fourth column and allow one to easily compare the two gliders.

only makes sense mod 4. Since we will be using $\frown$ distance in relation to $\bar{E}$s, we will understand $\frown$ distances to always be interpreted mod 4. In Figure 8, the second $\bar{E}$ is $\overset{3}{\frown}$ from the first.

For a $C_2$ glider, the associated column of ether triangles on each side is simply the column closest to the glider. We can describe the four relative positions of the $C_2$ and $\bar{E}$ in Figure 9 as being the cases where the $\bar{E}$ is $\overset{0}{\frown}$, $\overset{1}{\frown}$, $\overset{2}{\frown}$, and $\overset{3}{\frown}$ from the $C_2$.

Between $C$ gliders, we can measure an absolute $\frown$ distance, which we will indicate by writing it in small type between the $C$s, so for example $C_1\,{}^1C_1$ represents two $C_1$s that have exactly two vertical columns of ether triangles between them, each $C_1$ associating itself with the nearer column.

### 3.2.3 Preservation of Spacing

Looking at Figure 10, we see that it contains many collisions between $C_2$s and $\bar{E}$s. Each collision is of the form of the fourth collision in Figure 9, where the $\bar{E}$ is $\overset{3}{\frown}$ from the $C_2$, and the result is that a new $\bar{E}$ and $C_2$ get created. We can and will think of this collision as a $C_2$ and an $\bar{E}$ crossing each other.

Since the emerging $C_2$ from one of these collisions is $\overset{0}{\frown}$ from the $\bar{E}$ that produced it, and the next $\bar{E}$ must be $\overset{3}{\frown}$ from this emerging $C_2$ in order to cross it, we see that the second $\bar{E}$ must be $\overset{3}{\frown}$ from the first $\bar{E}$ if they are both to cross the "same" $C_2$.

Similarly, we can examine the left side of one of these collisions, and note that the $C_2$ leading to the collision is $\overset{3}{\frown}$ from the $\bar{E}$ it produces.

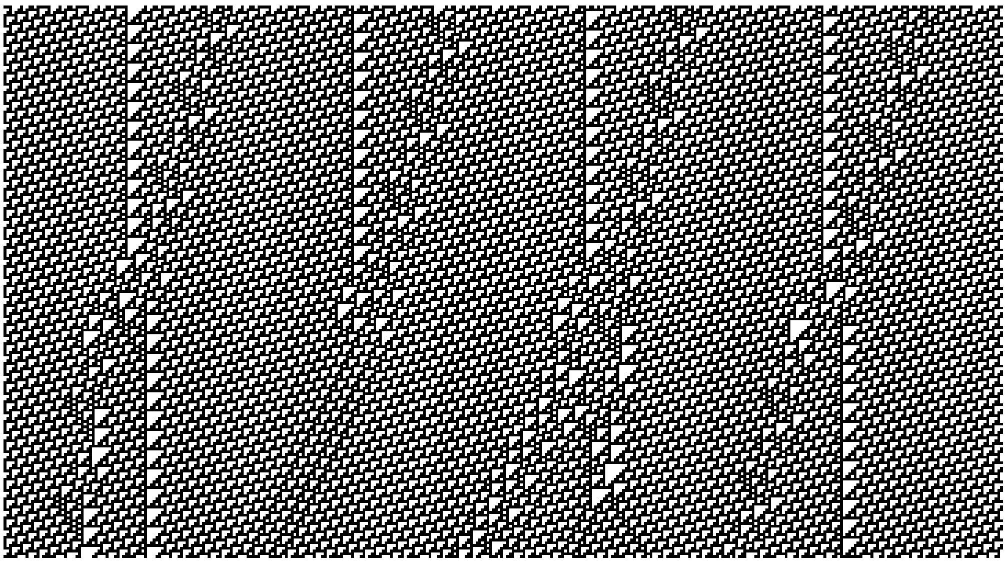

**Figure 9.** The four possible collisions between a $C_2$ and an $\bar{E}$.

Since we know this emerging $\bar{E}$ must be $\overset{3}{\wedge}$ from the next $C_2$ it will pass through, we can add these (mod 4) to find that two $C_2$s must be $\overset{2}{\wedge}$ from each other in order for an $\bar{E}$ to be able to cross them both.

So we know that if we have several $C_2$s spaced $\overset{2}{\wedge}$ from each other, followed by several $\bar{E}$s spaced $\overset{3}{\wedge}$ from each other, and the first $\bar{E}$ is $\overset{3}{\wedge}$ from the last $C_2$, then all of the initial collisions will be crossing collisions, as in Figure 10.

But what about the remaining collisions? We do not have much control over the gliders resulting from the earlier collisions, so how can we make sure the newly produced gliders will be aligned so as to cross each other? If we consider several $C_2$s crossing an $\bar{E}$, it is clear that each $C_2$ suffers exactly the same absolute displacement during its crossing, since the crossing collisions are identical, and so the positions of the $C_2$s *relative to each other* is perfectly preserved once they have all crossed the $\bar{E}$. This means that if they were $\overset{2}{\wedge}$ from each other to begin with, then they will still be $\overset{2}{\wedge}$ from each other after an $\bar{E}$ has crossed them all, and so a second $\bar{E}$, if it crosses the first $C_2$, will cross all the $C_2$s just like the first $\bar{E}$ did. So in fact the only possibility for all the remaining collisions is that they must all be crossing collisions!

In general, when several $C_2$s cross several $\bar{E}$s, as in Figure 10, the $C_2$s will end up with exactly the same spacing as they started with, and the same is true for the $\bar{E}$s. For example, in Figure 10, the $C_2$s start out in the form $C_2\,{}^{6}C_2\,{}^{10}C_2\,{}^{14}C_2$, and they end up in just the same arrangement.

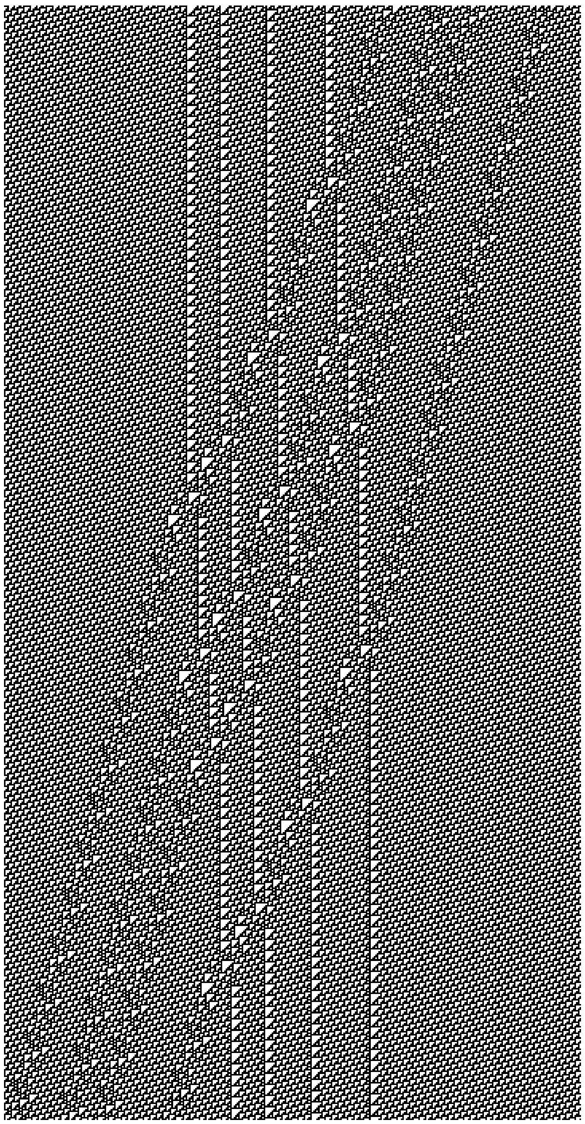

**Figure 10**. When $\bar{E}$s cross $C_2$s, the spacings are preserved, both between the $C_2$s, and between the $\bar{E}$s.

We do not need to do any calculations, or even know exactly how much displacement results from each crossing, to know that the relative spacings will be preserved. This phenomenon of spacing preservation provides a neat solution to one of the main obstacles in one dimensional computation: The problem of getting data to cross over other data. By representing data in terms of spacings between gliders of the same type, we can enable data travelling at two different speeds to cross without interference, all using just one single type of crossing collision.

If we were ever tempted to use different types of gliders to represent different values of data, we should disabuse ourselves of that conceit now that we see how this method of representing data only requires one type of crossing collision in order to let all values of data cross each other, whereas using different gliders for different values would require many different crossing collisions to happen to exist in phases compatible with each other in order for any transfer of data to be possible.

Now it is clear why our measurements all concern themselves only with the relative positions of neighboring gliders: Not only is that what determines which collisions will occur, but it is also the way we will represent data.

### 3.2.4 Data Conversion

In Figure 6, we saw three collisions in which an $A^4$ effectively converted an $\bar{E}$ into a $C_2$. Is it possible to use this as a way to convert data encoded with $\bar{E}$s into data encoded with $C_2$s?

Recalling that if two $\bar{E}$s cross a $C_2$, such as in Figure 10, then the second must be $\overset{3}{\frown}$ from the first, and that an $\bar{E}$ must also be $\overset{3}{\frown}$ from the $C_2$ it is going to cross, we see that the $C_2$ it is going to cross must always be $\overset{0}{\frown}$ from the preceding $\bar{E}$. Looking at the three collisions of Figure 6 which convert an $\bar{E}$ into a $C_2$, we see that for the first one, the resulting $C_2$ is $\overset{3}{\frown}$ from the $\bar{E}$ instead of $\overset{0}{\frown}$, meaning that the next $\bar{E}$ that comes along would not be able to cross it, so this collision cannot be used for data conversion. However, for the second and third $C_2$ producing collisions, the $C_2$ does happen to be $\overset{0}{\frown}$ from the $\bar{E}$, and so both of these collisions can work for data conversion. Based on their relative appearances, we will call the third one, where the $\bar{E}$ is $\nearrow_5$ from the $A^4$, the "short" collision, and we will call the second one, where the $\bar{E}$ is $\nearrow_2$ from the $A^4$, the "big" collision.

Let's consider the first collision shown in Figure 6, where the end result is that the $A^4$ and $\bar{E}$ are recreated on opposite sides of each other. Again, we can consider this as an $A^4$ crossing an $\bar{E}$. The $A^4$ is displaced significantly by the crossing, but like before, this will not concern us. If we want two consecutive $A^4$s to pass through an $\bar{E}$, we see that the first $A^4$ is $\nearrow_5$ from the $\bar{E}$ emitted from the first collision, which in turn must be $\nearrow_0$ from the next $A^4$ in order to pass through it, so the upper $A^4$ must be $\nearrow_5$ from the lower in order for them both to both be able to pass through an $\bar{E}$. We will in general assume that $A^4$s should be spaced like this, since that will allow us to send $\bar{E}$s through them if necessary.

Now we are in a position to better examine the spacings between $\bar{E}$s. If we consider two $\bar{E}$s, the second of which is $\nearrow_k$ from the first, and we suppose that the first one crosses an $A^4$, then since the $A^4$ emerging from that crossing is $\nearrow_1$ from the $\bar{E}$ above it, the second $\bar{E}$ will only be $\nearrow_{k-1}$ from that emerging $A^4$. This means that the relation between the two $\bar{E}$s, plus the knowledge that the first $\bar{E}$ will cross the $A^4$, is enough to tell us how the second $\bar{E}$ will hit the $A^4$. So for example if we want the second $\bar{E}$ to cross the $A^4$ just like the first one did, then since we want it to be $\nearrow_0$ from the $A^4$ in order to cross it, we will need it to be $\nearrow_1$ from the first $\bar{E}$. However, if we want the second $\bar{E}$ to be converted into a $C_2$ with the short reaction, then we will want it to be $\nearrow_5$ from the $A^4$ so as to cause a short reaction, and so we would need it to be $\nearrow_6$ (i.e. $\nearrow_0$) from the first $\bar{E}$.

Now let's consider the same question, but this time suppose that the first $\bar{E}$ is turned into a $C_2$ by the $A^4$ (say with the short reaction) instead of passing through it. So we have two $\bar{E}$s, the second of which is $\nearrow_k$ from the first, and we suppose that the first one is turned into a $C_2$ by a short reaction with an $A^4$. If the $\bar{E}$s are $\overset{3}{\frown}$ from each other then we know that the second $\bar{E}$ will pass through the $C_2$, and then if there is another $A^4$, we would like to know what collision will occur between it and the second $\bar{E}$. We will assume as before that the $A^4$s are $\nearrow_5$ from each other. But we have a problem, in that the ether used for comparing the two $\bar{E}$s is separated by the $C_2$ from the ether in which the emerging $\bar{E}$ will hit the $A^4$, so we can't immediately compare that $A^4$ to either of the original $\bar{E}$s.

Solving this problem turns out to be quite simple; we merely need to associate diagonal ether rows on one side of the $C_2$ with diagonal ether rows on the other side. To do this, we first notice that on both sides of the $C_2$, the diagonal rows used for $\nearrow$ measurements end at the $C_2$ next to one of the $C_2$'s large triangles. So let's associate each large triangle of the $C_2$ with the diagonal row that ends at it, both on the left and on the right. This gives us an association between rows on opposite sides of the $C_2$.

So now, we can combine known measurements that will answer our question. We know that the upper $A^4$ is $\nearrow_5$ from the lower $A^4$, and we can measure that the first $\bar{E}$ is $\nearrow_4$ from the upper $A^4$ as measured through the lower ethers through the $C_2$ below the first collision, and the second $\bar{E}$ is $\nearrow_k$ from the first $\bar{E}$, and we can measure that the $\bar{E}$ emerging from the crossing is $\nearrow_4$ from the incoming second $\bar{E}$ as measured through the ethers across the $C_2$ above the crossing, so in total, since the $C_2$ was crossed once each way, thus cancelling any offset it might have contributed, the emerging $\bar{E}$ is $\nearrow_{5+4+k+4}$, or $\nearrow_{k+1}$, from the lower $A^4$.

So for example, if we want the second $\bar{E}$ to be converted into a $C_2$ with a short reaction just like the first one was, then we want it to be $\nearrow_5$ from the lower $A^4$, which means it should just be $\nearrow_4$ from the previous $\bar{E}$. Or, if we want the second $\bar{E}$ to pass on through the $A^4$, then since it should be $\nearrow_0$ from the $A^4$, we conclude that it should be $\nearrow_5$ from the previous $\bar{E}$.

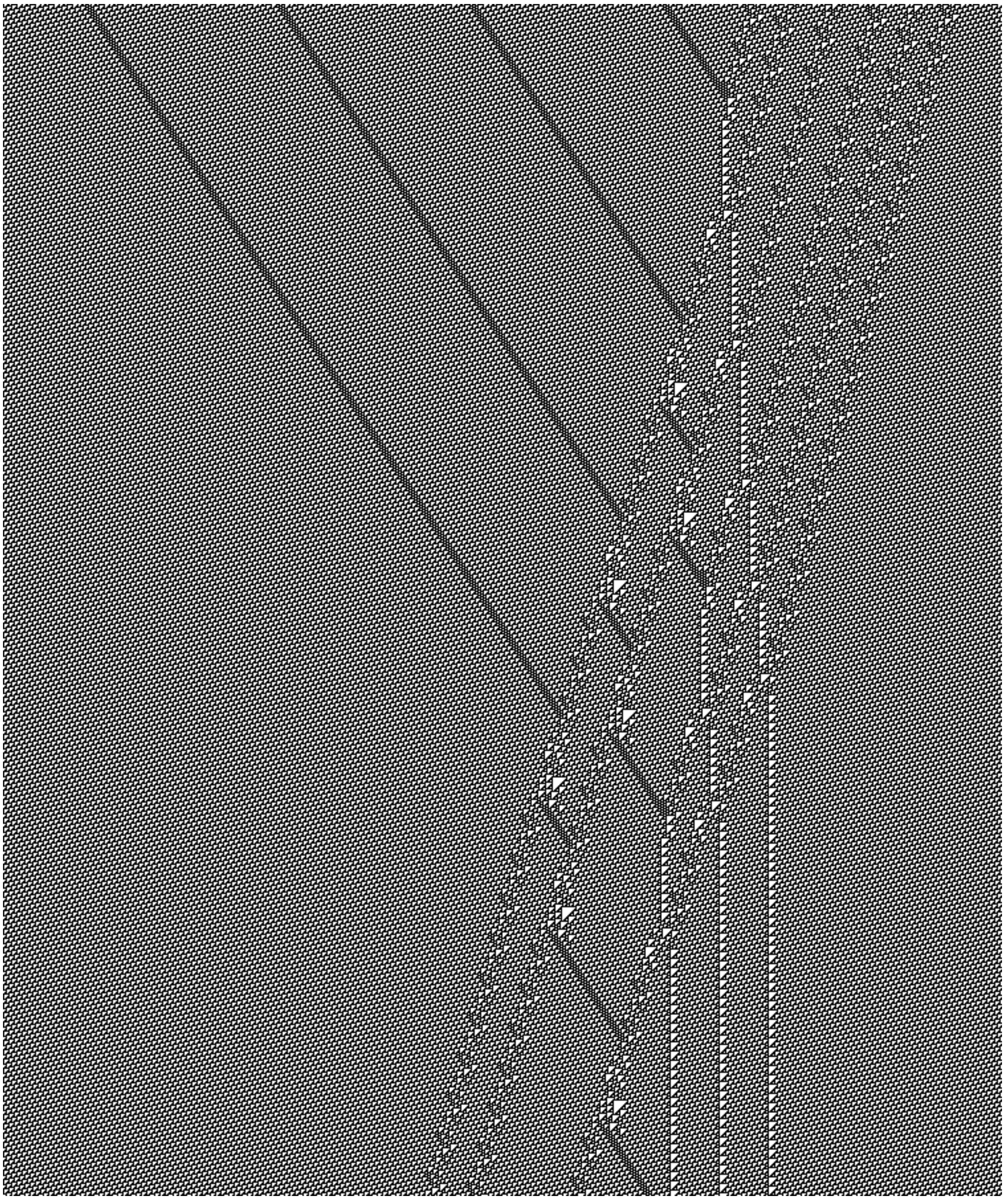

**Figure 11.** Assuming each $A^4$ is $\nearrow_5$ from the previous, then $\bar{E}$s which are $\overset{3}{\frown}$ from each other can either pass through all the $A^4$s, or be converted into $C_2$s, based solely on their relative $\nearrow$ distances from each other.

We can sum up some of our results on data transmission and conversion for future reference:

- If we space $C_2$s so that each is $\overset{2}{\frown}$ from the previous, then if an $\bar{E}$ crosses one, it will cross them all.
- If we space $\bar{E}$s so that each is $\overset{3}{\frown}$ from the previous, then if the first one crosses a $C_2$, then they all will.
- If we space $A^4$s so that each is $\nearrow_5$ from the previous, then if an $\bar{E}$ crosses one, it will cross them all.
- If an $\bar{E}$ crosses an $A^4$, the next $\bar{E}$ will too if it is $\nearrow_1$ from the first $\bar{E}$.
- If an $\bar{E}$ crosses an $A^4$, the next $\bar{E}$ will instead have a short reaction if it is $\nearrow_0$ from the first $\bar{E}$.
- If an $\bar{E}$ has a short reaction with an $A^4$, then the next $\bar{E}$ will also have a short reaction (with the next $A^4$) if it is $\nearrow_4$ from the first $\bar{E}$.
- If an $\bar{E}$ has a short reaction with an $A^4$, then the next $\bar{E}$ will instead pass through the next $A^4$ if it is $\nearrow_5$ from the first $\bar{E}$.

Now, as our final measurement task, we will examine how the spacings between $\bar{E}$s are converted by $A^4$s into spacings between $C_2$s. As we saw, if two neighboring $\bar{E}$s are going to be converted into two $C_2$s by short reactions with two properly spaced $A^4$s, then the second $\bar{E}$ must be $\overset{3}{\frown}$ and $\nearrow_4$ from the first. However, this still leaves some flexibility in the overall distance between the two, and this flexibility is what can be used to encode transmitted data.

We will assume that the spacing between the $A^4$s is fixed, and look at how a change in the $\bar{E}$ spacing affects the $C_2$ spacing. To examine changes in the $\bar{E}$ spacing, we will hold the first $\bar{E}$ fixed, and look at how we can move the second $\bar{E}$ while maintaining its $\overset{3}{\frown}$ $\nearrow_4$ relation with the first $\bar{E}$. Maintaining the $\nearrow_4$ relation is the same as maintaining the $\nearrow_0$ position with respect to the lower $A^4$, and any such position can be attained simply by moving the $\bar{E}$ back and forth in the $\vec{A}$ direction, so that the $A^4$ hits it sooner or later than it otherwise would. By moving the $\bar{E}$ one unit in the $\vec{A}$ direction, its $\frown$ distance from its neighbors is changed by one. So to preserve the $\overset{3}{\frown}$ relation which allows it to cross the $C_2$, we must move it by four $\vec{A}$ units at a time.

If we move the second $\bar{E}$ to the lower right by $4\vec{A}$, then the collision with the $A^4$ will occur exactly four $\vec{A}$ units to the lower right of its original position, and so the resulting $C_2$ will be produced exactly four columns to the right of its original position, closer to the first produced $C_2$. So widening the gap between the $\bar{E}$s leads to a narrower gap between the $C_2$s, and this gap between the $C_2$s can be adjusted in increments of four columns with this method. This makes sense, since we already knew that the short collision is guaranteed to create the second $C_2$ so that the first one is $\overset{2}{\wedge}$ from it, or in other words that only spacings of the form $C_2 \,{}^{4k+2}\, C_2$ are possible. Now we see that not only are possible spacings restricted to this form, but also conversely every spacing of this form is possible, within the maximum spacing allowed by the $A^4$s and the minimum spacing of $C_2 \,{}^{10}\, C_2$ obtainable from these collisions.

If instead of holding the $A^4$s fixed and varying the spacing between $\bar{E}$s, we consider the spacing between two $\bar{E}$s to be fixed and vary the lower $A^4$ by increments of $6\vec{B}$, we see that the location of the collision can be adjusted in increments matching the $\bar{E}$'s period, which again corresponds to being able to adjust the position of the $C_2$ in increments of four columns. So given some specific spacing between $\bar{E}$s that are $\nearrow_4$ and $\overset{3}{\wedge}$ from each other, and any particular goal of $C_2 \,{}^{4k+2}\, C_2$ with $k \geq 2$, it is always possible to have $A^4$s that are properly spaced to achieve this goal.

So now we see how information, encoded in spacings between $\bar{E}$s, can travel through $C_2$s, be converted into spacings between $C_2$s by $A^4$s, and remain intact while allowing more $\bar{E}$s to cross over.

## 4. The Construction

Now it's time to put everything together, finding explicit arrangements of gliders in Rule 110 to implement something along the lines of the universal glider system of Section 2.3.

### 4.1 The General Idea

In Figure 10 we saw how $\bar{E}$s and $C_2$s could cross over each other, so they seem like good candidates for representing moving data crossing tape data. And the ability of $A^4$s to convert the $\bar{E}$s into $C_2$s is just what we need for an ossifier. So our approach will be to use $C_2$s for tape data, $\bar{E}$s for moving data, and $A^4$s for ossifiers.

As suggested in Section 3.2.3, we will encode the *Y*s and *N*s of tape data and moving data by using different spacings between $C_2$s for tape data, and between $\bar{E}$s for moving data. The specific details of what spacings to use are flexible, so let's turn our attention to the right hand side.

On the right hand side, we still have to represent leaders and table data in some way, so that when a leader hits the tape, the result is either a rejector, wiping out the ensuing table data, or an acceptor, converting the table data into moving data.

With the help of a good nondeterministic oracle, we can quickly determine that the representations shown in Figure 12 will turn out to be useful. Figure 12 shows a sequence of eight $E$ gliders (mostly $\bar{E}$s) coming from the right, hitting four vertical $C_2$s. The four $C_2$s together

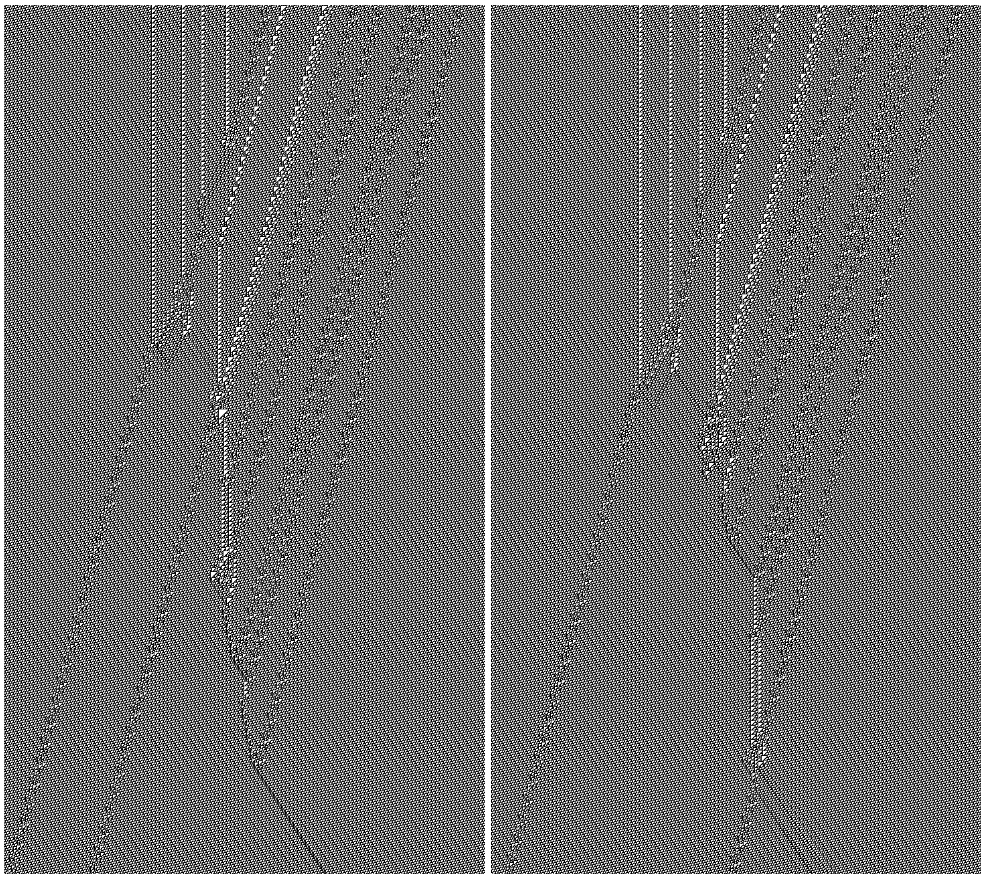

**Figure 12.** A character of tape data being hit by a leader. In the first picture, the leader hits an $N$ and produces a rejector $A^3$. In the second picture, a $Y$ is hit, producing an acceptor $A\,{}^4A\,{}^1A$. In both cases, two "invisible" $\bar{E}$s are emitted to the left. The first $\bar{E}$ of the leader reacts with the four $C_2$s in turn, becoming an invisible $\bar{E}$ at the end, and emitting two $A$s along the way. The difference in spacing between the center two $C_2$s in the two pictures, representing the difference between an $N$ and $Y$ of tape data, leads to different spacings between the two emitted $A$s. This causes the second $A$ to arrive to the $C_3$–$E^4$ collision at a different time in the two cases. In the first case, the $A$ converts the $C_3$ into a $C_2$ just before the collision, while in the second case, it arrives in the middle of the collision to add to the mayhem. The different outcomes are then massaged by the five remaining $\bar{E}$s so that a properly aligned rejector or acceptor is finally produced.

represent one character of tape data, and the eight $E$s together form a leader. To the lower right, either a rejector, $A^3$, or an acceptor, $A\,{}^4A\,{}^1A$, is emitted.

But, disturbingly, in each case, two $\bar{E}$s have been emitted to the left, heading into the rest of the tape. As it will turn out, they are in the phase where they will not only pass through the tape, but also pass right through the $A^4$s that would ossify normal moving data. We will call them *invisibles*, since they pass through both the tape data and the ossifiers with no effect.

One nice thing about invisibles is that since they never go away, they constitute a permanent record of everything that has happened. Since each acception or rejection generates a pair of invisibles, with the gap between them indicating whether it was an acception or rejection that generated them, the invisibles encode a complete history of the activity of the cyclic tag system that is being emulated.

If we continue questioning our oracle, we can quickly find the promising *components* shown in Figure 13. These components have the nice property that they are erased by a rejector but turned into two $\bar{E}$s by an acceptor. The rejector or acceptor then continues on, ready to reject or accept more components, until finally, the rejector or acceptor will be absorbed by the next leader as shown in Figure 14, converting that leader from a *raw leader* into the *prepared leader* form originally shown in Figure 12.

For reasons that will become clear later, the first component after a leader is always a *primary component*, and all the rest are *standard components*. Two components together represent one character of table data. The spacing between them determines which character: A wider spacing is used to represent an $N$, and a narrower spacing is used to represent a $Y$. If accepted, the two components produce four $\bar{E}$s, representing one character of moving data. Again, the spacing between the middle two $\bar{E}$s determines which character is represented by the moving data. An ossifier consists of four $A^4$s, and converts the four $\bar{E}$s of a character of moving data into four $C_2$s, representing a new character of tape data. This process inverts the spacings, so now a narrow gap between the middle two $C_2$s indicates an $N$, while a wider gap indicates a $Y$, as seen in Figure 12.

So now we see the overall form of the construction. But we do not yet see why it should actually work — when two gliders collide, they can do so in many ways, and there is no reason to blindly assume that we will be able to arrange the parts described above so that the collisions will all work out as promised.

Luckily, our oracle was wise, and it will all work out. The remaining parts of this section will present the details of how this works.

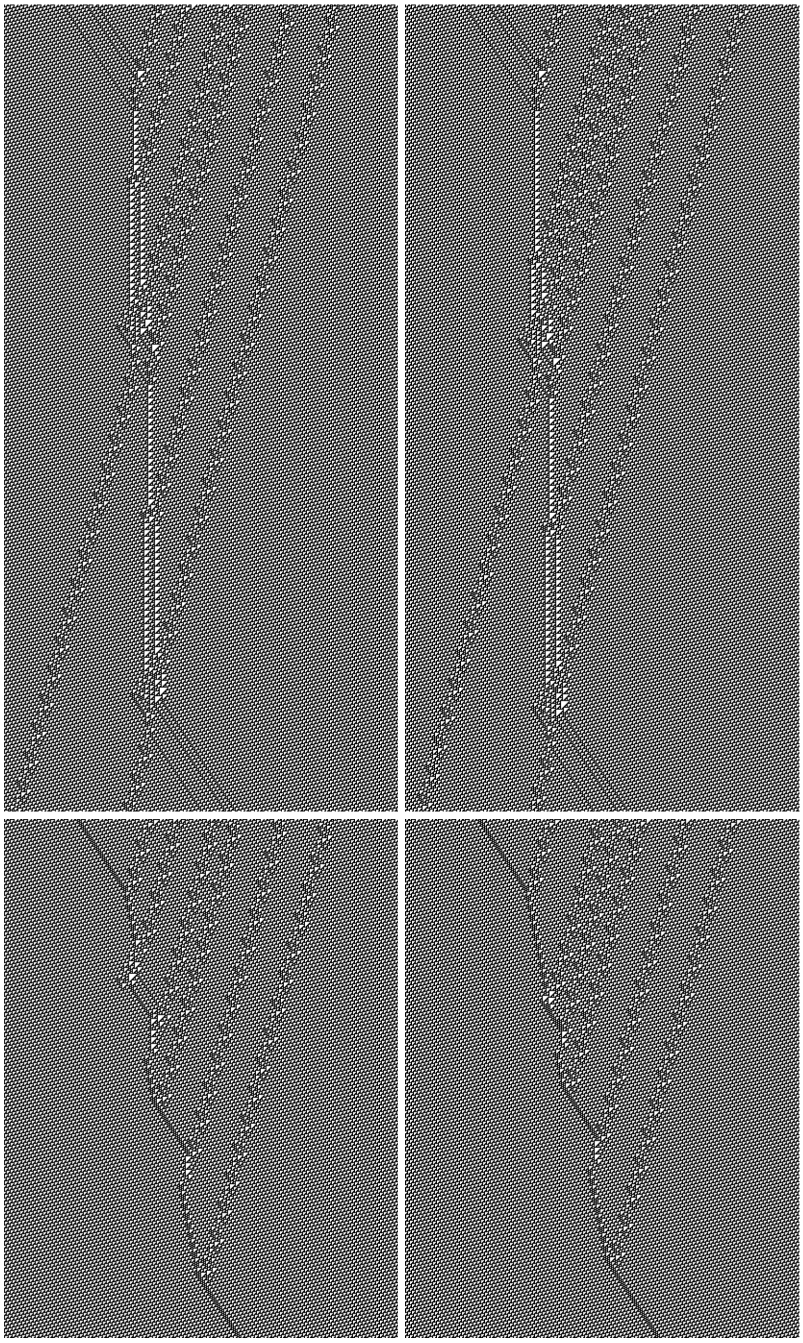

**Figure 13.** Components getting accepted or rejected. The left pictures show primary components; the right pictures show standard components. The upper pictures show acception; the lower pictures show rejection.

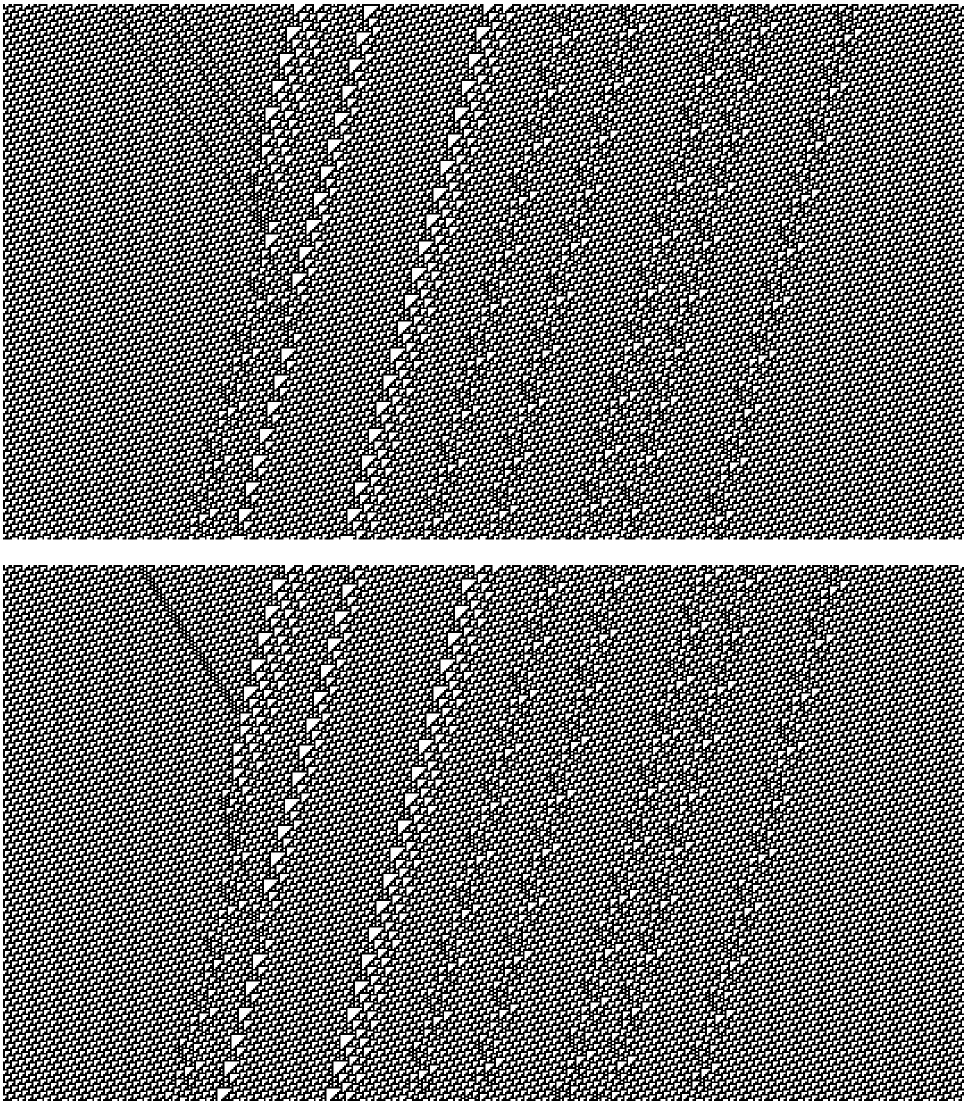

**Figure 14.** Both an acceptor and a rejector are absorbed by a raw leader, which becomes a prepared leader in the process.

## 4.2 Details of the Middle

In the middle, we have moving data crossing to the left over tape data. As long as the $C_2$s are spaced $\overset{2}{\wedge}$ from each other, and each $\bar{E}$ is $\overset{3}{\wedge}$ from the first $C_2$ it's supposed to pass through, we know that all the collisions will be "crossing" collisions, just like in Figure 10.

For completeness, we also need to be sure that the gliders are spaced far enough from each other that the crossing collisions will not bump into each other. For the $C_2$s, that means they must be spaced no closer than $C_2\,{}^{6}C_2$. For the $\bar{E}$s, there isn't any real danger, since if there is any ether between them, then they are able to both cross a $C_2$.

So the middle will work as planned if the following conditions are met:

- The $C_2$s must be spaced $\overset{2}{\frown}$ from each other.
- Adjacent $C_2$s must be spaced no closer than $C_2\,{}^{6}C_2$.
- Each $\bar{E}$ must be $\overset{3}{\frown}$ from the first $C_2$ it crosses.

## 4.3 Details of the Left

On the left, moving data is turned into tape data by the ossifiers, while invisibles pass through the ossifiers unharmed. We will use the "short" reaction from Section 3.2.4, rather than the "big" one, for ossification.[9]

Like in Section 3.2.4, using the short reaction to convert an $\bar{E}$ into a $C_2$ guarantees that the $C_2$ will be spaced $\overset{2}{\frown}$ from the last one that the $\bar{E}$ went through, with a minimum spacing of $C_2\,{}^{10}C_2$, so we see that the first two conditions listed for the middle to work correctly will always be satisfied, since all tape data will arise from this reaction, except for the tape data present in the initial conditions when the system is started, which can of course be explicitly arranged so as to satisfy the conditions.

Two consecutive ossifiers (two groups of four $A^4$s) must have a gap of the form $A^4\,{}^{6k+5}A^4$ between them, with $k$ large enough to satisfy the spacing condition of Section 2.3. This requirement is easily satisfied, since the spacings between the $A^4$s can be set directly with the initial conditions.

The very first (rightmost) $A^4$ of the first ossifier must pass through the very first $\bar{E}$ that it hits, since that will be the first invisible from the first tape symbol being read. Since in this case there is no previous $\bar{E}$, we cannot say that the $\bar{E}$ will be hit correctly based on its relation to a previous $\bar{E}$. However, we can easily make sure that it is hit correctly just by explicitly setting up the initial conditions so that the ossifiers are correctly aligned for this collision.

So here on the left, all that remains to be shown is that the tape data characters that are created by ossification will have the correct spacings to properly represent $Y$ and $N$ characters. As shown in Figure 12,

[9]The construction can also be attempted using the "big" reaction, but then it turns out to be impossible to adjust the gliders so that the calculations in Section 4.4.1 can work out correctly, and so such a construction cannot work. This emphasizes the importance of checking such details whenever a "construction outline" is proposed, before believing that the construction will actually work as claimed.

a tape data $Y$ is represented by $C_2\,{}^{18}C_2\,{}^{18}C_2\,{}^{14}C_2$, and a tape data $N$ is represented by $C_2\,{}^{18}C_2\,{}^{10}C_2\,{}^{14}C_2$. As discussed in Section 3.2.4, reducing the central $C_2$ spacing from 18 to 10 can be accomplished by increasing the central $\bar{E}$ spacing by $8\vec{A}$. Since the first and third $C_2$ spacings are constant, the third and first $\bar{E}$ spacings should be constant.

As far as the collisions not bumping into each other, it is clear from Figures 12 and 13 that the $\bar{E}$s will be spaced far enough apart that they will have no problem, except possibly the last $\bar{E}$ of some moving data could be close to the first invisible from the following leader, so we should make sure those aren't super close to each other. Since the $\bar{E}$s will not be too close to each other, the $A^4$s will have to be fairly far apart in order to generate the small $C_2$ spacings, so there won't be any danger of the $A^4$ collisions bumping into each other.

So, our conditions for proper operation of the left hand side are:

- An invisible $\bar{E}$ following a previous invisible $\bar{E}$ must be $\nearrow_1$ from it.
- A moving data $\bar{E}$ following an invisible $\bar{E}$ must be $\nearrow_0$ from it.
- A moving data $\bar{E}$ following a previous moving data $\bar{E}$ must be $\nearrow_4$ from it.
- An invisible $\bar{E}$ following a moving data $\bar{E}$ must be $\nearrow_5$ from it.
- The moving data for a $Y$ and $N$ should be the same, except that the central spacing for an $N$ should be $8\vec{A}$ larger.
- The last $\bar{E}$ of moving data shouldn't be super close to the following invisible.

## 4.4 Details of the Right

The right hand side is where all the trickiness lies. We must arrange everything so as to satisfy all the constraints required by the middle and the left. Luckily, we have one thing to our advantage: When adjusting the leaders and components, we can adjust their $\nearrow$ and $\frown$ distances independently. To see this, note that if we move something vertically by one $C$ period, i.e. the distance from one triangle in a vertical ether column to the next one, then we have not affected its $\frown$ distance, but we have changed its $\nearrow$ distance by 1. Conversely, if we move something diagonally by one $A$ period, i.e. the distance from one triangle in a diagonal ether row to the next one, then we have not affected its $\nearrow$ distance, but we have changed its $\frown$ distance by 1. So by using these two methods of adjustment, we can clearly get any combination of $\nearrow$ and $\frown$ distances.

In light of this, we will consider the $\nearrow$ and $\frown$ distances separately.

Even when the $\nearrow$ and $\frown$ distances are specified for a leader or component, there is still some flexibility in its location. Using this flexibility as described in Section 3.2.4, we can easily satisfy the last two conditions listed for the left side to work. In general, we will not abuse the flexibility; we will be consistent in our spacing so that the end result is a periodic sequence encoding the cyclic appendant list.

#### 4.4.1 $\nearrow$ Distances on the Right

As discussed in Section 3.2.4, an invisible $\bar{E}$ must be $\nearrow_1$ and $\overset{3}{\frown}$ from a previous invisible, and Figure 12 shows that this is indeed the relation of the second invisible to the first, in both the acception and the rejection cases.

Since the primary and standard components are the same except for the first two spacings in their $\bar{E}$s, they yield exactly the same spacing between the two emitted $\bar{E}$s when accepted. This means that the first and third spacings are always the same in the $\bar{E}$s for a character of moving data. The central spacing can be increased or decreased by $8\vec{A}$ by moving the entire second component $8\vec{A}$ farther or closer to the first. Such a motion will clearly not affect the component's interaction with the acceptor or rejector — the entire interaction will move by the same amount, and the emitted acceptor or rejecter will be in exactly the same position as it otherwise would.

We know that a moving data $\bar{E}$ must be $\nearrow_4$ from a previous moving data $\bar{E}$. Between the $\bar{E}$s of an accepted component, Figure 13 shows us that this will be the case. But how do we know whether this will be the case between consecutive $\bar{E}$s generated by neighboring components? We can tell by comparing both $\bar{E}$s to the acceptor which connects the components. In both primary and standard components, the emerging acceptor is $\nearrow_0$ from the second $\bar{E}$ of emitted moving data. And in the ensuing component, which will always be a standard component, the first emitted $\bar{E}$ is $\nearrow_4$ from the incoming acceptor. Putting these together, we see that the first $\bar{E}$ emitted by the second component will always be $\nearrow_4$ from the previous $\bar{E}$ of moving data. So now we see that consecutive $\bar{E}$s of moving data will always have the second being $\nearrow_4$ from the first.

Note that in order for the acceptor signal to hit a standard component correctly, the first $\bar{E}$ of the component must be $\nearrow_5$ from the acceptor. Since the acceptor is itself $\nearrow_3$ from the last $\bar{E}$ of the previous component, a standard component must always be placed $\nearrow_2$ from the previous component in order for acception to work correctly. Fortunately, the corresponding calculation based on a rejector also indicates that a standard component must be $\nearrow_2$ from the previous component in order for rejection to work correctly, so this placement works for both acception and rejection.

Now all we have left to check is the $\nearrow$ distances between moving data and invisibles, and between components and leaders.

Let's look at the first component after a leader, which is always a primary component. In the case of acception, the acceptor is $\nearrow_0$ from the last invisible, and the first $\bar{E}$ of moving data emitted from the primary component is in turn $\nearrow_0$ from the acceptor, so the moving data is a total of $\nearrow_0$ from the invisible, which is just right.

If we look at how the primary component should be placed relative to the leader, we see that the first $\bar{E}$ of the primary component must be $\nearrow_5$ from the acceptor, which is in turn $\nearrow_3$ from the last $\bar{E}$ of the leader, so the primary component should be placed a total of $\nearrow_2$ from the leader. Again, the same calculation using the rejector gives the same requirement, so placing the primary component $\nearrow_2$ from the leader will work for both acception and rejection.

Now let's look at the last component before a leader. If the component was accepted, then we know the acceptor is $\nearrow_0$ from the last emitted $\bar{E}$ of moving data. The first $\bar{E}$ of the prepared leader will be $\nearrow_0$ from the acceptor, so it will also be $\nearrow_0$ from the last $\bar{E}$ of moving data. But this is not the end, because what we really need is the $\nearrow$ distance from the last $\bar{E}$ of moving data to the first invisible that will eventually be emitted when the leader hits a character of tape data, and this relation must be measured in the ether to the left of the character of tape data that will be read.

To do this, we will use the same method of measuring "across" the $C_2$s as we used in Section 3.2.4. Since the first and third spacings in the character of tape data are fixed, the relationship between the first $\bar{E}$ of the prepared leader and the temporary $\bar{E}$ that it becomes after interacting with two of the $C_2$s is fixed, as is the relationship between that temporary $\bar{E}$ and the invisible $\bar{E}$ that it becomes after interacting with the other two $C_2$s. For now let's say that the first $\bar{E}$ of the prepared leader is $\nearrow_k$ from the previously emitted $\bar{E}$, which may have been either an invisible or moving data. So, measuring across the $C_2$s, we see that the invisible is $\nearrow_2$ from the temporary $\bar{E}$, which is $\nearrow_1$ from the first $\bar{E}$ of the prepared leader, which we are saying is $\nearrow_k$ from the previous $\bar{E}$, which is $\nearrow_2$ from its position after crossing one $C_2$, which is $\nearrow_2$ again from its position after crossing the second $C_2$, which is $\nearrow_2$ yet again from its position after crossing the third $C_2$, which is $\nearrow_2$ again from its position after crossing the fourth $C_2$. So in total, the invisible is $\nearrow_{2+2+2+2+k+1+2}$, or $\nearrow_{k+5}$ from the previous $\bar{E}$.

Now, if the previous $\bar{E}$ was moving data, as we were assuming before this big calculation, then $k$ is 0, and the invisible will be $\nearrow_5$ from the previous $\bar{E}$ of moving data, which is just right.

On the other hand, if the previous components were rejected, then the previous $\bar{E}$ was itself an invisible from back before all the components got rejected, and we need to see what $k$ would be in this case. As we see in Figure 14, the leading $\bar{E}$ of the prepared leader is $\nearrow_2$ from the rejector, which in turn as we see in Figure 13 is $\nearrow_5$ from its previous position for every standard component it rejects, and then $\nearrow_1$ from its position before the primary component, which is in turn $\nearrow_4$ from the previous invisible. So if there were $c$ characters of table data, then there were $2c$ components, of which one was primary, so we have $k = 4 + 1 + 5(2c - 1) + 2 = 4c + 2$. Since an invisible should be $\nearrow_1$ from a previous invisible, we also need to have $k + 5 = 1$, or $k = 2$, since $k$ is a value mod 6. Equating these expressions for $k$ yields $2 = 4c + 2$, or $0 = 4c$, mod 6, which is satisfied only if $c$ is a multiple of 3.

This might startle us, but we'll have to accept the fact that invisibles will only be lined up correctly after rejection if the number of characters in the table data was a multiple of three. Luckily, it will turn out that it is no problem to require that all table data entry lengths must be multiples of three, as we will show in Section 4.5.

There are still a couple of issues to wrap up concerning the $\nearrow$ distances on the right. One is that we need to be sure that the raw leader can be positioned after the final component so that both the acceptor and rejector will prepare it correctly. Since the raw leader does not start with an $\bar{E}$, it is not immediately obvious how to measure its position compared to the final component, perhaps in part because its alignment must depend on more than just the leading $E^5$, since the $\bar{E}$ that is produced by the acceptor or rejector must be properly aligned not only with the neighboring $E^1$, but also with the more distant $\bar{E}$s, which have twice the period of $E^n$s. In other words, one has to be careful that the initial $\bar{E}$ doesn't wind up one $E^n$ period, or half an $\bar{E}$ period, off from where it should be.

Our solution to this problem may seem almost natural by now: We will associate the diagonal ether rows on opposite sides of an $E^n$ by associating its large size 8 and size 5 triangles to each other and to the ether so that triangles are associated if the jagged lower right edge of one touches the upper left corner of the next. This gives an association for every third diagonal row of ether, and the association is trivially extendable to every row.

Using this association, we can verify that in each of the pictures in Figures 12 and 14, the second $\bar{E}$ of the prepared leader (after the $E^4$) is $\nearrow_0$ from the prepared leader's first $\bar{E}$, as measured through the $E^1$ and $E^4$.

We can also see that in the case of acception, the first $\bar{E}$ of the raw leader should be $\nearrow_3$ from the acceptor, which is in turn $\nearrow_3$ from the last $\bar{E}$ of the last component, so the raw leader must be placed so that its first $\bar{E}$ is $\nearrow_0$ from the last $\bar{E}$ of the preceding component, as measured throught the $E^n$s. In the case of rejection, the raw leader's first $\bar{E}$ should be $\nearrow_0$ from the rejector, which is in turn $\nearrow_0$ from the last component's last $\bar{E}$, so fortunately we get the same requirement for the placement of the raw leader in either case.

Finally, it must be pointed out that our formula $k = 4c + 2$ for the $\nearrow$ distance between a previous invisible and the leading $\bar{E}$ of a prepared leader is wrong in the case that there are no components at all, since then there is no primary component and no standard component, whereas

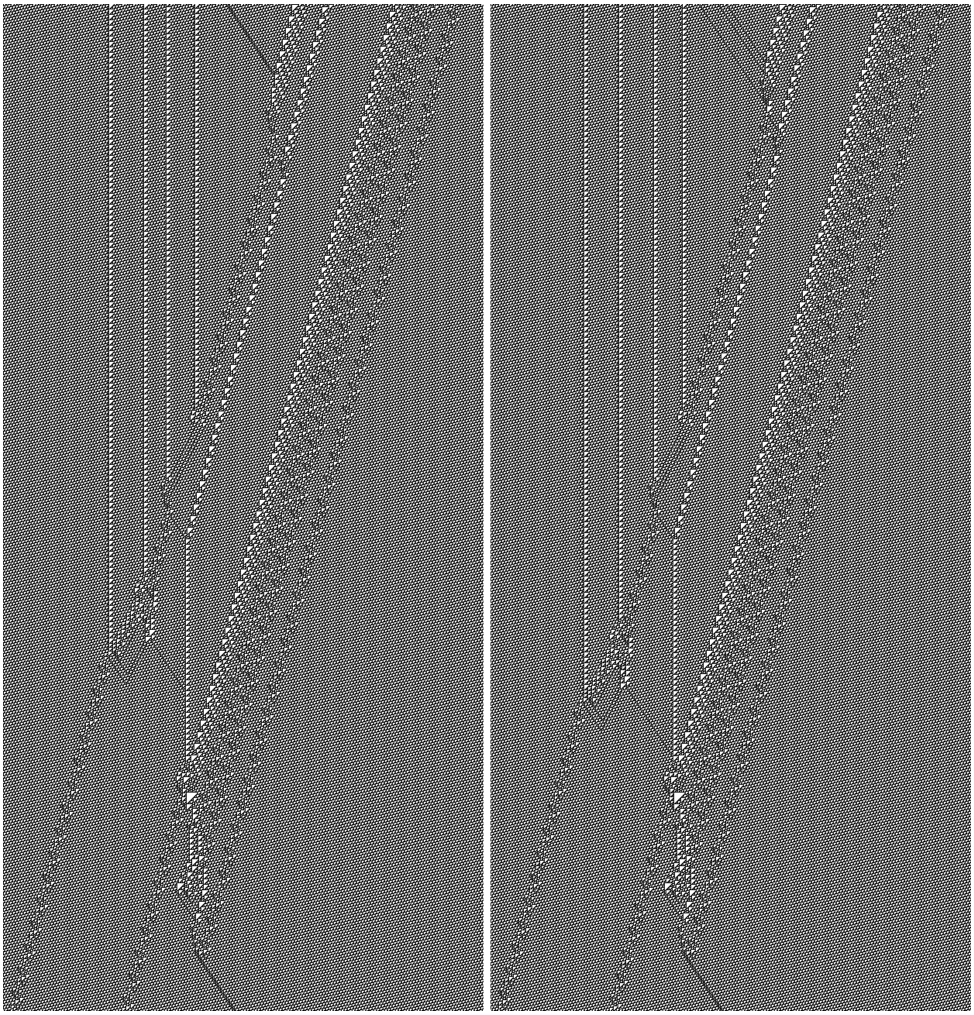

**Figure 15.** The left picture shows a short leader absorbing a rejector and then hitting an $N$ of tape data. The right picture shows a short leader absorbing an acceptor and then hitting a $Y$ of tape data. Even with the wide spacing of the $Y$'s $C_2$s, the second $A$ still turns the $C_3$ into a $C_2$ just before the $E^4$ hits it, so from that point on, the pictures are the same, and only three $\bar{E}$s are needed to turn the signal into a properly aligned rejector.

our formula treated it as being one primary component, *minus* one standard component, which doesn't make sense. Since the formula is incorrect, the spacing is also incorrect, and so it turns out that we need to use a different kind of leader when it will not be followed by any components, which is the case corresponding to an empty appendant.

Figure 15 shows a *raw short leader* absorbing a rejector in exactly the same way a regular leader does, becoming a *prepared short leader* and then continuing on to hit a $Y$ of tape data, which yields exactly the same result as if it hits an $N$ of tape data, since if the appendant is empty we don't need to worry about distinguishing a $Y$ from an $N$. The short leader always emits a rejector, $\nearrow_0$ from the last invisible. Since the next raw leader (either short or normal) will be placed so that it is prepared correctly by this rejector, the leading $\bar{E}$ of its prepared form will be $\nearrow_2$ from the rejector and therefore also $\nearrow_2$ from the previous $\bar{E}$, which is just what the formula wanted it to be all along, so the next invisible will be correctly positioned relative to the last invisible from the short leader.

In summary, we see that it is possible to arrange the right hand side so that the $\nearrow$ distances throughout the system will be correct, regardless of which appendants turn out to get accepted or rejected.

### 4.4.2 $\frown$ Distances on the Right

Now we will examine the $\frown$ distances and see if we can place the leaders and components so that the $\frown$ distances always turn out as intended. This analysis will be somewhat simpler than that for $\nearrow$ distances, since our main concern with $\frown$ distances is just that each $\bar{E}$ must be $\overset{3}{\frown}$ from the first $C_2$ it crosses, which is the same thing as being $\overset{3}{\frown}$ from the previous $\bar{E}$ which crossed the $C_2$.

When a component emits two $\bar{E}$s, as in Figure 13, we can see that the second is always $\overset{3}{\frown}$ from the first. The first one's relation to whatever the previous $\bar{E}$ was will be determined by the placement of the component.

Since the components can only be compared above the acceptor, and the emitted $\bar{E}$s can only be compared below the acceptor, to relate the two we will need a method of associating the ether columns above an $A^n$ with the columns below. This is easy, since such columns are perfectly aligned with each other anyway, and the vertical columns of dots in a large $A^n$ help keep it clear which column is which if they are far apart. With this association of ether columns across $A^n$s, crossing an $A^n$ clearly yields the same associated columns as crossing $n$ $A$s.

We can see that for both primary and standard components, the first emitted $\bar{E}$ is $\overset{2}{\frown}$ from the first $\bar{E}$ of the component, and for primary components, standard components, and leaders (but not short leaders), the last $\bar{E}$ of the leader or component is $\overset{1}{\frown}$ from the last emitted $\bar{E}$.

Where should we place a component so that the emitted $\bar{E}$s will be correctly aligned during acception? If we place it so that the first $\bar{E}$ is $\overset{k}{\frown}$ from the previous $\bar{E}$ of the previous component or leader, then the first $\bar{E}$ emitted from the component is $\overset{2}{\frown}$ from the first $\bar{E}$ in the component, which is $\overset{k}{\frown}$ from the last $\bar{E}$ of the previous component or leader, which is $\overset{1}{\frown}$ from the previously emitted $\bar{E}$, so in total the component's first emitted

$\bar{E}$ is $\overset{2+k+1}{\frown}$ from the previously emitted $\bar{E}$. Since it needs to be $\overset{3}{\frown}$ from the previously emitted $\bar{E}$, we see that $k$ should be 0, so all components should be placed $\overset{0}{\frown}$ from the previous component or leader.

The only remaining kind of $\bar{E}$ which needs to be aligned correctly to pass through the tape data is the first invisible produced by a leader or short leader. Since the first invisible produced by a leader is always $\overset{1}{\frown}$ from where the leftmost $C_2$ of the character of tape data read by the leader was, and that $C_2$ is $\overset{2}{\frown}$ from the $C_2$ to its left, the first invisible is always $\overset{3}{\frown}$ from the first $C_2$ it needs to cross, as required.

Now we know that all the $\bar{E}$s which need to pass through the tape data will be positioned correctly to do so. But there is one very important case where an $\bar{E}$ should *not* be positioned so as to pass through the tape data: The first $\bar{E}$ of a prepared leader must be $\overset{1}{\frown}$ from the first $C_2$ that it hits, so that the collision will occur as shown in Figures 12 or 15. This means it must also be $\overset{1}{\frown}$ from the previously emitted $\bar{E}$.

If the previously emitted $\bar{E}$ was the second invisible generated by a short leader, then since the short leader's last $\bar{E}$ is $\overset{2}{\frown}$ from the second invisible as measured through the rejector it always produces, the first $\bar{E}$ of the following prepared leader should be $\overset{3}{\frown}$ from the last $\bar{E}$ of the short leader, as measured back through the rejector, in order to be a total of $\overset{1}{\frown}$ from the second invisible.

But if instead the previously emitted $\bar{E}$ was a final $\bar{E}$ of moving data, then since the last component's last $\bar{E}$ was $\overset{1}{\frown}$ from the final $\bar{E}$ of moving data as measured through the acceptor, the first $\bar{E}$ of the prepared leader should be $\overset{0}{\frown}$ from the last component's last $\bar{E}$, as measured back through the acceptor, in order to be a total of $\overset{1}{\frown}$ from the last emitted $\bar{E}$.

The only remaining case is that the last $\bar{E}$ emitted prior to the prepared leader was the second invisible of a regular leader, in which case that leader must have produced a rejector which wiped out the table data. Using the alignment information deduced from considering acception, we will track the $\frown$ distances down through the components to see how the prepared leader will relate to the last $\bar{E}$ in this case, and we hope to find that it will be aligned correctly.

The last $\bar{E}$ of the previous leader is $\overset{1}{\frown}$ from the second invisible it emits, and the first $\bar{E}$ of the first component is $\overset{0}{\frown}$ from that. For both a primary component and a standard component, the final $\bar{E}$ is $\overset{3}{\frown}$ from the first $\bar{E}$ as measured through the rejector at one end and back through the rejector at the other end, and as we saw earlier, each component's first $\bar{E}$ is $\overset{0}{\frown}$ from the previous component's last $\bar{E}$. The $\bar{E}$ of the prepared leader must be $\overset{0}{\frown}$ from the last $\bar{E}$ of the last component as measured through the rejector, since that's where it was when we measured it through an acceptor, and both methods of associating columns across three $A$s are the same. So in total, the first $\bar{E}$ of the prepared leader must be $\overset{1+3k}{\frown}$ from the last emitted invisible, if there were $k$ intervening rejected components. Since we were needing this to be $\overset{1}{\frown}$, we see that $k$ must be a multiple of 4. Since there are two components for each character of table data, we see that there must be an even number of characters of table data in order for everything to be fine.

As with our previous finding that the number of characters must be a multiple of three, this will turn out not to be a problem, as the next section will show.

### 4.5 Divisibility Requirements

As we saw in the course of analyzing our construction, the number of characters in each appendant of the cyclic tag system being emulated must be both a multiple of three and a multiple of two, so in other words it must be a multiple of six.

This will be automatically satisfied if the number of symbols in the tag system being emulated by the cyclic tag system is a multiple of six. We can easily add extra symbols to a tag system without changing its behavior in any way, so we see that this constraint poses no problem at all.

### 4.6 Some Undecidable Questions for Rule 110

As discussed in Section 2.3, given a fixed repeating pattern to the left and right, it is undecidable whether a given finite initial condition will lead to periodicity in Rule 110's behavior.

We also explained that it could be undecidable whether a certain glider ever gets produced, so let's look at how that works. If an ossifier does hit a character of tape data, the collision of an $A^4$ with a $C_2$ always produces an $\bar{E}$ and an $A$. The $A$ will turn the neighboring $C_2$ into a $C_1$, while the $\bar{E}$ will continue until it is hit by the next $A^4$, which due to its spacing from the first $A^4$ will necessarily turn the $\bar{E}$ back into a $C_2$. The third $A_4$ will hit that $C_2$, again producing an $\bar{E}$ and an $A$, and this time the $A$ will turn the neighboring $C_1$ into an $F$.

So another specific example of an undecidable question for Rule 110 is: Given an initial middle segment, will there ever be an $F$?

Let us stop here.

## References

[1] Stephen Wolfram. *A New Kind of Science* (Wolfram Media, 2002).

[2] Stephen Wolfram originally introduced his numbering scheme in *Statistical Mechanics of Cellular Automata*, in *Reviews of Modern Physics*, volume 55, pages 601-644 (July 1983).

[3] John Conway wrote an exposition of "The Game of Life" (a now famous two dimensional cellular automaton that he discovered), including

a sketch of how one might build a universal computer within it, in *Winning Ways For Your Mathematical Plays, Volume 2*, by E. Berlekamp, J. Conway, and R. Guy (Academic Press, 1982).

[4] Alvy Ray Smith III showed in *Simple Computation-Universal Cellular Spaces*, in the *Journal of the ACM*, volume 18, number 3, 1971, pages 339-353, that there are universal one dimensional cellular automata, and he gave examples with $(k = 40, r = .5)$, $(k = 18, r = 1)$, $(k = 11, r = 1.5)$, $(k = 7, r = 2.5)$, $(k = 5, r = 3.5)$, $(k = 4, r = 4)$, $(k = 3, r = 6)$, and $(k = 2, r = 10)$, using the $(k, r)$ notation of [14], where $k$ is the number of states a cell may have, and $r$ is the "radius" of the neighborhood (not including the cell itself), so the update function has $2r + 1$ arguments.

[5] Jürgen Albert and Karel Culik II exhibited a universal one dimensional cellular automaton with $(k = 14, r = 1)$, in *A Simple Universal Cellular Automaton and its One-Way and Totalistic Versions*, in *Complex Systems*, volume 1, 1987, pages 1-16. (Interestingly, the cover of that first issue featured a picture of Rule 110's activity.)

[6] Kristian Lindgren and Mats G. Nordahl exhibited a universal one dimensional cellular automaton with $(k = 7, r = 1)$ and $(k = 4, r = 2)$, in *Universal Computation in Simple One-Dimensional Cellular Automata*, in *Complex Systems*, volume 4, 1990, pages 299-318.

[7] Alan Turing presented what are now called Turing machines in 1936, and he was the first to exhibit a universal Turing machine. In 1956, Shannon proposed looking for universal Turing machines with a minimal *states* × *symbols* product, and Ikeno found a $10 \times 6$ one in 1958. Then Watanabe found an $8 \times 6$ one in 1960. Minsky found a $7 \times 6$ one in 1960, followed by Watanabe finding an $8 \times 5$ one in 1961, after which Minsky found a $6 \times 6$ one in 1961 and a $7 \times 4$ one in 1962. In 1979, Rogozhin presented $24 \times 2$, $11 \times 3$, $7 \times 4$, $5 \times 5$, $4 \times 6$, $3 \times 10$, and $2 \times 21$ machines.

Based on Rogozhin's machines, which all use the same underlying idea, one can conjecture that a numerical measure of (*states* − 1) × (*symbols* − 1) is perhaps a little more accurate. This measure also reflects the fact that universal Turing machines are impossible with only one state or one symbol.

All of the above machines had a way of halting, whereas the Turing machines presented here do not, since they emulate Rule 110, which, being a cellular automaton, does not have an explicit mode of halting. However, the machines presented here are certainly capable of "effective computation" in the sense of Minsky, and so questions about their behavior, such as "Will this sequence of symbols ever appear on the tape?", are undecidable.

Another difference is that the above machines start with a tape which, outside of the finite "initial condition" region, is "blank" (periodic with period one, with the same symbol used on both the left and right). The Turing machines presented here require longer (but fixed) periodic patterns on the left and right. However, as the last paragraph of chapter 5 in [10] makes clear, this is not much of a conceptual change.

[8] Thanks to David Eppstein for figuring out that the two-symbol machine can be achieved with only seven states. (personal communication, 1998)

[9] Alonzo Church's "effective calculability", Alan Turing's "computability", Emil Post's "canonical systems", Stephen Kleene's "general recursive functions", Raymond Smullyan's "elementary formal systems", and many other people's precise ideas for describing "what can be calculated", have all turned out to have exactly the same calculational capability. This surprising phenomenon lead to the thesis, argued for by many of the people involved as well as others, and generally accepted, that these systems are capable of carrying out any specifiable procedure whatsoever.

[10] Chapters 6 of Marvin Minsky's book *Computation: Finite and Infinite Machines* (Prentice-Hall, 1972) provides a very readable and motivated (by chapter 5) introduction to Turing machines.

[11] Emil Post studied tag systems as a graduate student in 1921, and found that it was very hard to prove anything at all about them.

[12] John Cocke gave essentially this proof in *Universality of Tag Systems With* $P = 2$ in the *Journal of the Association for Computing Machinery*, volume 11, number 1, 1964, pages 15-20. (Minsky gave the same proof in part 14.6 of [10].) While following the same basic idea, the proof presented here is simpler, requiring just 10m symbols instead of 16m (Minsky) or 17m (Cocke, although m of them exist just for clarity and are never read), and only requiring two "passes" over the tape instead of three, per Turing machine step. The first proof of universality in tag systems was a rather complicated proof, for tag systems with $p$ a product of at least two distinct primes (e.g. 6), given in *Recursive Unsolvability of Post's Problem of "Tag" and Other Topics in Theory of Turing Machines* by Marvin Minsky in *Annals of Mathematics*, volume 74, number 3, 1961.

The observation that Cocke's proof approach works for Turing machines using $k$ symbols, if a tag system with $p = k$ is used, appears to be new. (The point of Cocke's paper was to show that tag systems with $p$ as low as 2 could be universal, since they had already been shown by Minsky to be universal for many higher values of $p$.)

[13] Table 15 in the appendix (pages 485-557) of *Theory and Applications of Cellular Automata*, by Stephen Wolfram (World Scientific, 1986), or on pages 575-577 of *Cellular Automata and Complexity* by Stephen Wolfram (Addison-Wesley, 1994), is devoted to Rule 110.

The table of "particles" (gliders) appearing there, due to Doug Lind, corresponds to the following: $\{\bar{B}, B, G^1, \bar{E}, E^1, F, E^2, C_3, C_1, C_2, D_1, D_2, A\}$ Due to some mistake, the $E^2$ is listed out of order, and with an incorrect period. If we remove it, then the list is a complete list of the commonly occurring gliders, with extendable ones listed just in their base form.

The final sentence says, "One may speculate that the behavior of rule 110 is sophisticated enough to support universal computation."

[14] Stephen Wolfram suggested that "class 4" cellular automata might be capable of universal computation in *Universality and Complexity in Cellular Automata*, in *Physica D*, volume 10, 1984, pages 1-35.

[15] Stephen Wolfram notes that some simple cellular automata are "class 4", but does not yet notice any as simple as $(k = 2, r = 1)$, in *Undecidability and Intractibility in Theoretical Physics*, in *Physical Review Letters*, volume 54, 1985, pages 735-738.