Glider Gun Guidelines

Stephen Wolfram
The Institute for Advanced Study, Princeton NJ 08540.

(March 1985)

In his *Computer Recreations* column in the May 1985 issue of *Scientific American*, A. K. Dewdney described some of the research that I have been doing on one-dimensional cellular automata. As he mentioned, there are some interesting questions about these systems that you may be able to help me answer. The questions do not rely on sophisticated mathematical knowledge, but to make progress on them will probably take a lot of work and some clever ideas. These notes define some of the questions, describe what I have worked out so far about them, and suggest some approaches you could take to them.

The central issue is what kinds of structures can occur in certain simple one-dimensional cellular automata. Although their basic rules are very simple, the overall behaviour of such cellular automata seems very complicated. I would like to be able to characterize just how complicated they are by identifying the structures and processes that can occur in them.

Many cellular automata give spacetime patterns that seem very chaotic, and in which one cannot really identify definite persistent structures. However, there are some cellular automata that I have called "class 4" in which definite localized structures occur. These cellular automata are comparatively rare, making up about 5% of the possibilities (with $k<5$ and $r<2$ - see below). There are various kinds of structures one can look for:

Periodic structures. These consist of localized patterns that remain unchanged with time, or follow a cycle, periodically returning to the same form. Usually these and other structures occur on a background of zeroes; but for some rules the background may have another state, or perhaps itself be a periodic "wallpaper" pattern.

Propagating structures or "gliders". These are localized patterns that systematically shift across the cellular automaton lattice with time. They may have a variety of speeds.

Glider guns. In the strictest definition, these are finite-sized patterns which periodically emit gliders. Including these gliders, such objects give rise to a pattern with a systematically increasing total size. One may in general look for structures in class 4 cellular automata that

increase in total size in this way, not necessarily just by emission of gliders. (Note however that in class 3 cellular automata such configurations are commonplace: the question is of interest only for class 4 cellular automata, such as those listed below.)

Self-reproducing structures. These would be like simple mathematical idealizations of living systems. A finite pattern in a class 4 cellular automaton would grow with time, and eventually yield a copy of itself. (Trivial self-reproduction phenomena can occur in rules that obey a "linear superposition" principle, according to which the patterns produced from a particular initial state correspond to simple superpositions of patterns produced say with a single cell seed.)

But ultimately the most interesting question is probably whether certain simple one-dimensional cellular automata are capable of "universal computation", so that they can emulate the operation of any computer. There is a rather complicated one-dimensional cellular automaton, with 18 states per site ($k=18$), that has been specifically constructed to imitate a universal Turing machine (a standard mathematical computer model), and so is capable of universal computation. But from empirical observations I strongly suspect that many much simpler cellular automata, such as those listed below, are also capable of universal computation. In no case, however, has this been proved.

One can prove that a cellular automaton is capable of universal computation by showing that it can imitate some other system that is already known to be a universal computer. For such a cellular automata it must be possible to find a particular initial state that evolves just like any of the states of the known universal computer.

There are various kinds of universal computers known. A good introduction to some of them is the book by F. S. Beckman, entitled "Mathematical Foundations of Programming", published by Addison-Wesley in 1981. One can minimize the amount of work needed to prove a cellular automaton computationally universal by choosing to show equivalence to the closest system already known to be a universal computer.

One possibility might be to show equivalence to a Turing machine. This consists of a memory tape divided into squares each containing one of a fixed set of symbols, together with a "head" or CPU which can move backwards and forwards on the tape, reading and writing symbols, according to a fixed set of internal rules. I think that the simplest known universal Turing machine has 4 possible symbols on the tape, and 7 internal states in its head. In a cellular automaton, the tape symbols should be represented by stable (or perhaps periodic) separated structures. The head would be represented by a structure consisting of a complicated sequence of site values, which has appropriate interactions with the tape symbol structures. The most direct transcription of the simplest known Turing machine yields the cellular automaton with 18 states mentioned above (see A. R. Smith, "Simple computation-universal cellular spaces", *Journal of Association for Computer Machinery*, volume 18, page 331 (1971)). But by allowing the tape symbols and the head to be represented by complicated structures, rather than just single site values, one should be able to imitate this Turing machine with a simpler cellular automaton. Indeed, if the cellular automaton is to be a universal computer, such an equivalence must always exist, but the necessary encoding could be extremely complicated.

A second and perhaps better possibility is to show equivalence to a standard digital computer. This was the approach used to show that the two-dimensional cellular automaton known as the "Game of Life" is capable of universal computation. One should be able to follow the outline of this proof quite closely. One needs to identify objects in the cellular automaton that act as wires (which carry signals), memories, logical gates, and probably a clock. For the rules discussed below, various of these objects have been found. But more are needed. Memories correspond to isolated periodic structures. Signals are propagated by gliders, which move periodically across the cellular automaton. These should interact with each other, and with memories, to implement various logical functions (such as "and", "nand", etc.). Given "gates" corresponding to various logical functions, one must find out how to arrange initial conditions for the cellular automaton to emulate an arbitrary digital circuit. Glider guns will probably also be needed to act as sources of

signals, and clocks.

For some further details on these questions, and on cellular automata in general, you can look at some of my published papers. Two that were written for a reasonably general audience are:

"Cellular automata", *Los Alamos Science* (Fall 1983) (available from Los Alamos National Laboratory, Los Alamos, NM 87545)

"Cellular automata as models of complexity", *Nature*, volume 311, page 419 (October 4, 1984).

Somewhat more technical papers are:

"Statistical mechanics of cellular automata", *Reviews of Modern Physics*, volume 55, page 601 (July 1983)

"Universality and complexity in cellular automata", *Physica D*, volume 10, page 1 (1984).

The latter reference has also appeared in a book entitled "Cellular automata", edited by D. Farmer, T. Toffoli and S. Wolfram, published by North Holland/Elsevier (1984).

A somewhat more technical paper, describing in particular some of the algorithms discussed below is:

"Computation theory of cellular automata", *Communications in Mathematical Physics*, volume 96, page 15 (1984).

Cellular automaton rules

Here are some class 4 cellular automaton rules to look for various kinds of structures in. Each of these rules is, I suspect, capable of universal computation, but it would be very interesting to prove this.

1. $k=3$, $r=1$, totalistic rule code 357:
   a diversity of periodic structures have been found, and I have recently found one rather complicated glider.
2. $k=2$, $r=2$, totalistic rule code 20:
   periodic and propagating structures are known, but no glider gun has been found, and interactions between the structures have not been studied.
3. $k=3$, $r=1$, totalistic rule code 792: periodic and propagating structures are known, but no glider gun has been found.
4. $k=2$, $r=3$, totalistic rule code 88 (Park's rule):
   periodic and propagating structures, together with a glider gun, have been found. These must now be assembled to make a universal computer.
5. $k=2$, $r=1$, rule number 193:
   periodic and propagating structures have been found, and some of their interactions have been studied. In the other rules, the "background" consists of cells in state 0. Here there is a rather elaborate "wallpaper" background, with period 14 horizontally and 7 vertically.

Notes

Tables and illustrations of the structures found in these rules are given at the end.

All phenomena should be investigated on lattices large enough that edge effects are irrelevant.

The values of k given specify the number of possible states (labelled say 0 through $k-1$) in the cellular automata. Different states can be represented by different colours or characters. The values of r give the "range" of the cellular automaton rule. When $r=1$, the new state of a particular cell depends only on its immediate neighbours to the left and right. When $r=2$, it also depends on the next neighbours on each side, and so on.

The "totalistic" rule codes used in cases 1 through 3 are defined in my September 1984 *Scientific American* article. "Totalistic" rule codes are defined in my September 1984 *Scientific American* article. For example, the $k=2$, $r=2$ cellular automaton with code 20 (case 2 above) has the following rule. First, find the numerical sum of the states of a site, and its two neighbours on the left and on the right. If this sum (whose maximum value is 5) is exactly 2 or 4, then make the new cell have state 1; otherwise make it have state 0. Remember that the sum is of *old* cell states; new states that have just been computed must be kept separate.

The rule for case 5 is of a different kind. First find the binary equivalent of 193: this is 11000001. Then the ith digit (starting from 0 on the right) gives the new state of a cell when the binary number formed by the three old cell states is exactly i. So if the three cells are 000, 110 or 111, the new cell has states 1; otherwise it has state 0.

Some other rules

Here are some more class 4 cellular automaton rules to look at:

$k=5$, $r=1$, totalistic code 53955: background of 1's.

$k=5$, $r=1$, totalistic code 522809355: period 2 background.

$k=5$, $r=1$, totalistic code 55135.

$k=4$, $r=1$, totalistic code 1004600.

$k=2$, $r=2$, totalistic code 52.

Search procedures

I know of two basic methods that can used to search for structures in one-dimensional cellular automata. The first is simply to test each possible initial configuration in turn, and observe its fate. The second is to use an algorithm (described below) that can systematically find all structures with a particular period in a given cellular automaton.

It is not very easy to find complicated structures in cellular automata. Systematic methods tend to take very much computer time. The tables given below each took many hours of CPU time on a reasonably fast computer (I mostly used a Ridge 32 - comparable in speed to a larger VAX). For example, the glider in the first table took about two days of CPU time to find. But I was making the computer do all the work. And I suspect that if I had spent enough time studying the patterns that the cellular automaton makes, then I would have been able to go a long way towards constructing interesting structures myself, without having to make the computer just search through all the possibilities. I think it is probably best to combine experimentation with some thinking, and some systematic computer searching.

When one does an exhaustive search of all possible initial configurations, say up to some given size, the same structures will be produced many times. One can tell that a definite structure has been produced by seeing whether the configurations produced are periodic in time. Then one must find out whether the structure has been produced before. This can be slightly tricky, because one may catch it at different points in its cycle on different occasions. One must also take account of gliders, which repeat themselves, but shifted over. My program always looks only at a "window" on the cellular automaton whose edges are just r cells away from the edges of the patterns produced, and which shift when the pattern shifts. Another problem is that some initial configurations may evolve to several separated structures, each of which has been seen before. One needs to make the program recognize these structures. One awkward case involves composites with gliders going in opposite directions, or gliders together with periodic structures. In these cases, the pattern never becomes periodic as a whole, even though its parts do.

The procedure just described can be used in principle to find all the structures that a particular cellular automaton can generate. One can test each possible initial configuration in turn. Or one can try initial configurations at random.

An alternative procedure is to use a systematic algorithm that finds all possible structures with a particular period in a given cellular automaton. Of course, as the tables show, some of the structures can have a pretty long period, and this algorithm really becomes impractical for anything but quite short periods. I will first describe how this algorithm works for finding configurations that are stable under a cellular automaton rule, and so remain unchanged from one time step to the next.

1. Write down the state obtained by applying the cellular automaton rule to each possible block of $2r+1$ cell states. Then discard those blocks in which the state of the centre cell changes. Only those blocks that remain are allowed to appear in the blocks that remain. Now one must simply find out what configurations (if any) can be made by stringing these allowed blocks together, with the blocks overlapping by $2r$ sites. For efficiency, it is good at this point to make a table that specifies which block can follow which other block in a configuration. (Such tables are equivalent to what are known as a "de Bruijn" graph.) Thus for example, 101 can be followed only by 010 and 011 (though one or both of these may not be among the set of blocks for which the centre cell is mapped to itself).
2. Now construct a "tree" of all the possible strings of blocks. Assume that in the "background", all cells have state 0. Then start with the block that represents the background (e.g. 000 for an $r=1$ rule). Let this block be the root of a tree. Then make branches to nodes corresponds to the allowed blocks that can follow this block. Continue this procedure until (a) no allowed blocks can follow the current block, or (b) the background block is reached again. In case (b), the sequence of blocks on the path starting at the root corresponds to a periodic configuration in the cellular automaton (the actual configuration can be read off from the sequence of centre cell states in the blocks).

To look for structures with period p greater than one, just form blocks of length $2rp+1$ that represent the effect of p applications of the cellular automaton rule, then use the algorithm outlined above. This algorithm can also be used to search for gliders of a predetermined speed: make the blocks give the results of the cellular automaton rule shifted by the approprite amount.

Epilogue

That is about all I can tell you about these questions. The rest is up to you. If you find something interesting, please do send me a letter, at the address given above, and send a copy to A. K. Dewdney, care of *Scientific American*.

I have printed up a set of six colour postcards of cellular automaton patterns. They are available from me at $2 (US) for each set. But I will certainly send a free set to anyone who finds an interesting new cellular automaton structure. And if enough of these are found, I will make a catalogue of them later this year. If you would like a copy of this, please send me a note.

Good luck!

Tables of structures

Structures with periods followed by L or R are left and right-moving gliders, respectively. Glider guns are indicated by G.

A code for the shortest initial configuration ("seed") that yields each structure is given. The code consists in the sequence of cell states for the configuration, treated as digits of a base k number, and given as a decimal number. (So, for example, configuration 29 with $k=3$ is $102 = 3^3+2$.)

period	seed
48	28
19	7795
19	8083
26	1706588
41L	4803890
41R	12269314

Structures found for the $k=3$, $r=1$ rule with code 357. (Searched up to size 14.)

period	seed
2	151
9R	187
1	189
22	195
9L	221
1R	635
1L	889
38	125231
4	595703
4	610999
4	624623

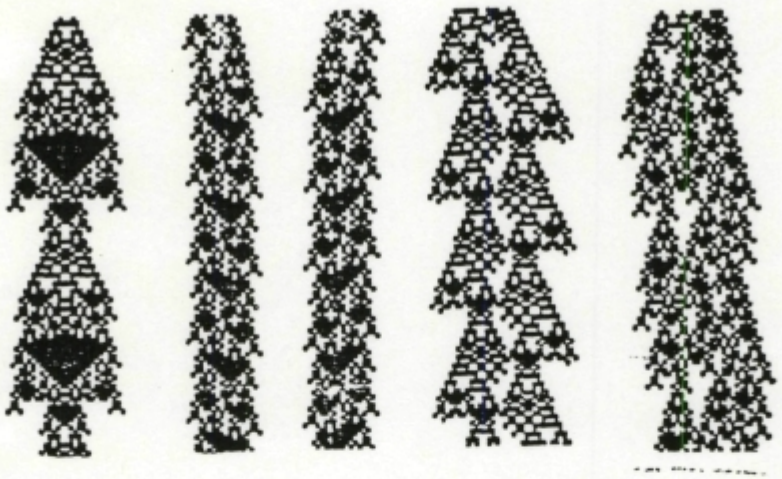
Structures found for the $k=2$, $r=2$ rule with code 20. (Searched up to size 21.)

period	seed
1	4
20	5
3R	101
3L	113
16	1625
3L	4187
3R	4561
3R	5252
3L	5540
12	12031
32L	35996
32R	40424
6R	141872
6L	148424

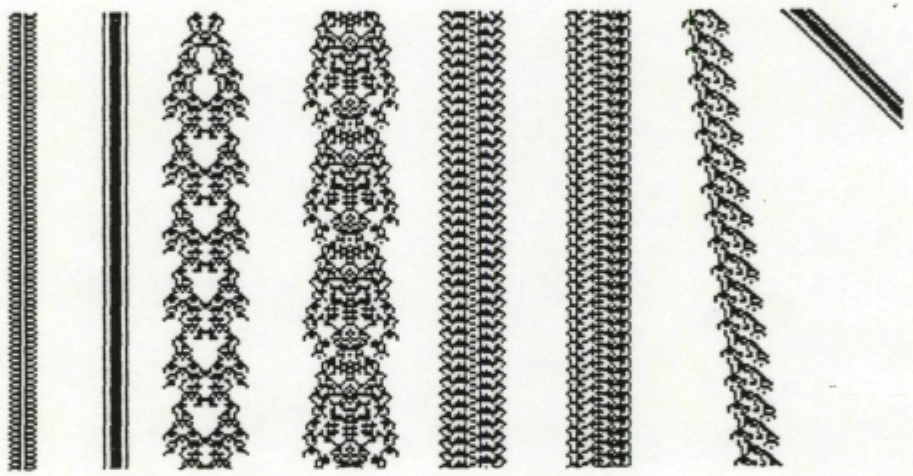
Structures found for the $k=3$, $r=1$ rule with code 792. (Searched up to size 11.)

period	seed
3	7
1R	207
1L	243
2	1631
2	3295
1	3579
3R	6591
3L	7135
238G	7167
34	1704415
3L	57307736539103
3R	69268037164555

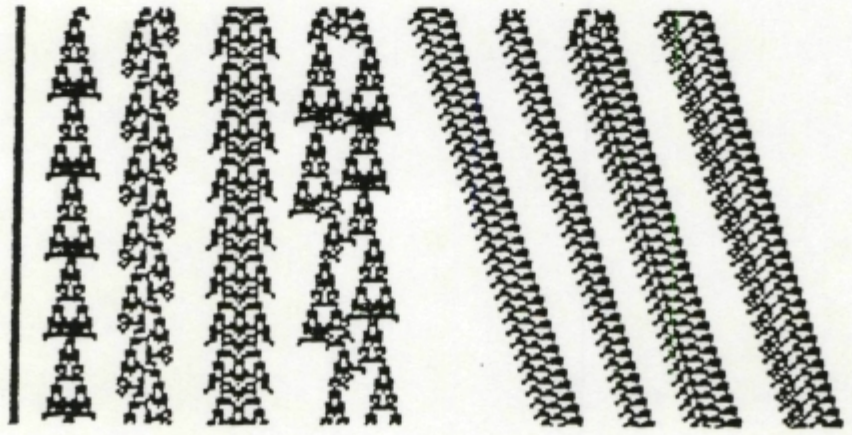
Structures found for the $k=2$, $r=3$ rule with code 88, by James Park of Princeton University. (All possible structures with periods up to 3 are now known; those not listed in the table have the form 11010000011110111111111101000(11011111110111011111000)*1110111111111011111 where the starred sequence can be omitted or repeated any number of times.)



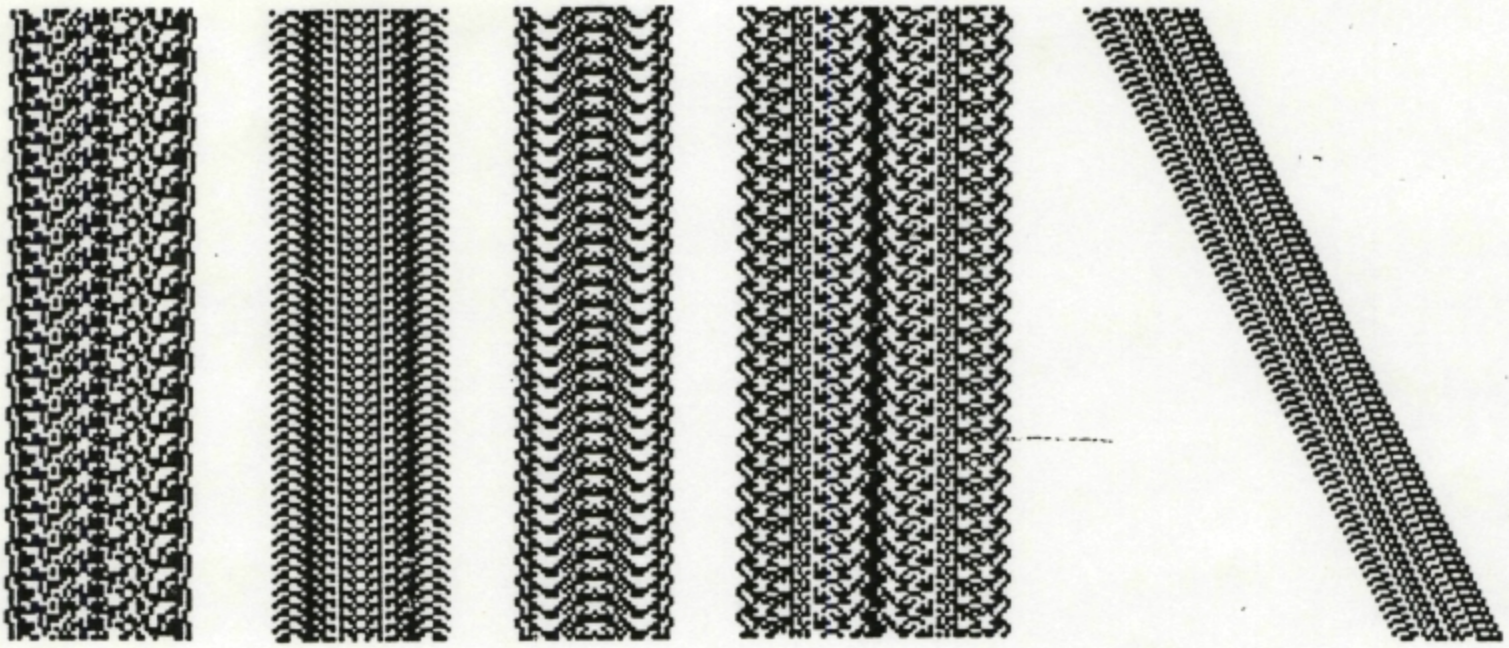
Structures found for the $k=3$, $r=1$ rule with code 357. (All nonzero sites shown black.) Can you find other gliders? Or a glider gun?



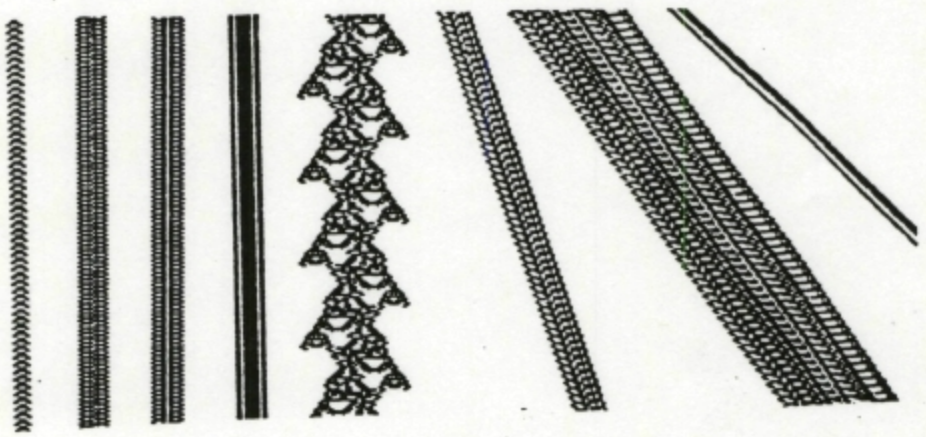
Structures found for the $k=2$, $r=2$ rule with code 20. Can you find a glider gun?



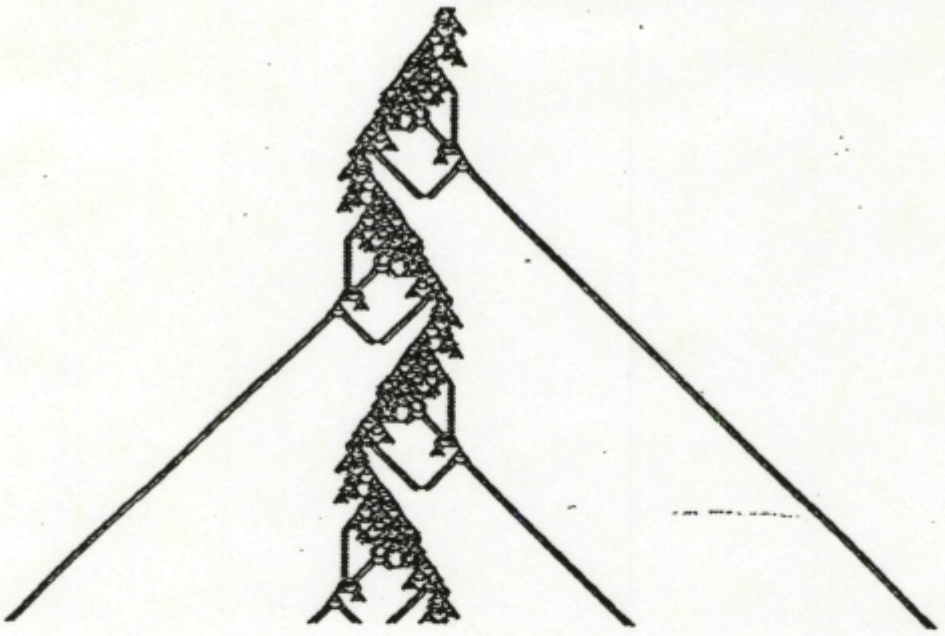
Structures found for the $k=3$, $r=1$ rule with code 792. (All nonzero sites shown black.) Can you find a glider gun?



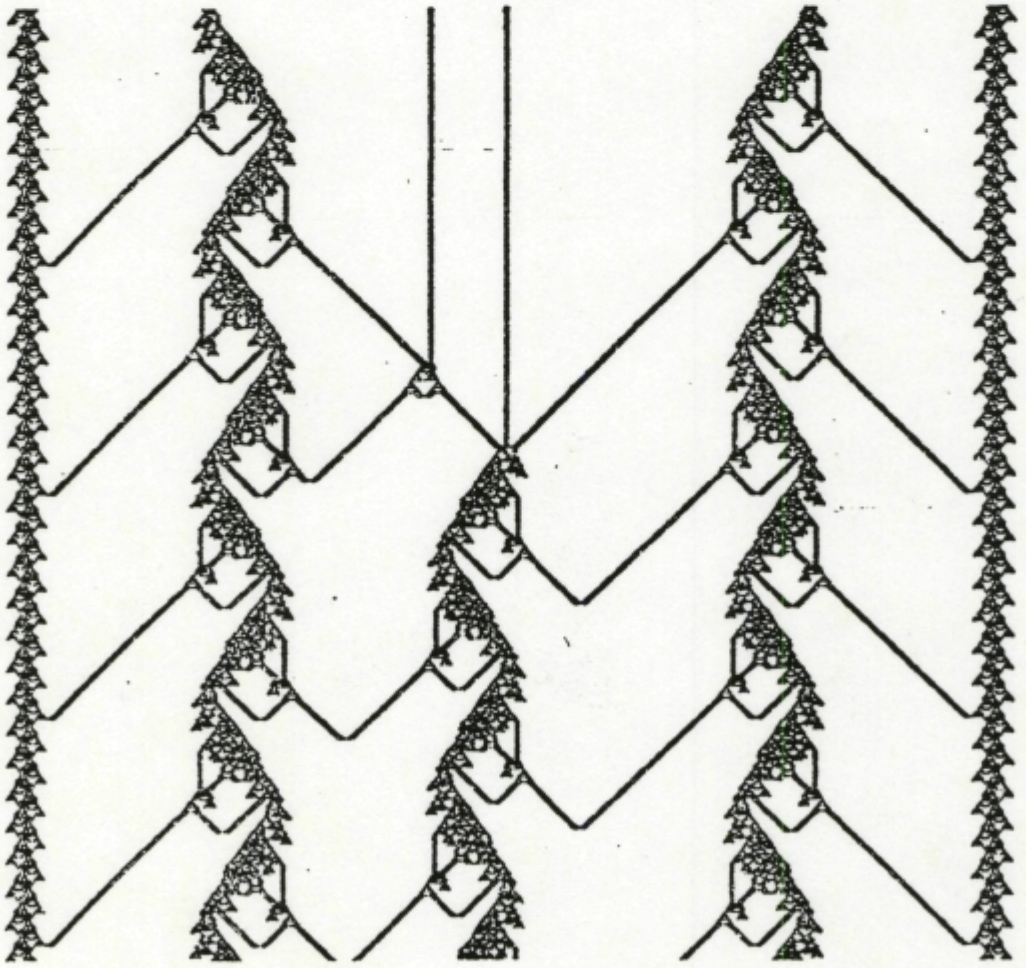
Some additional structures found for the $k=2$, $r=2$ rule with code 20 by Peter Grassberger, a physicist at the University of Wuppertal, West Germany.



Some structures found for the $k=2$, $r=3$ rule with code 88, by James Park of Princeton University.



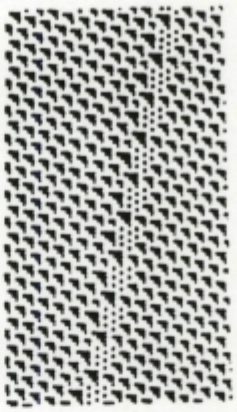
A glider gun found for the $k=2$, $r=3$ rule with code 88, by James Park.



Interaction between some structrures in the $k=2$, $r=3$ rule with code 88. The structures on the sides absorb all gliders that hit them. And in the centre, two simple initial structures combine with gliders from the two glider guns to create a new glider gun. (Picture courtesy of James Park.)

$s = \frac{-2}{3}$

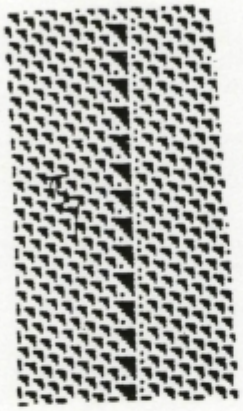
$s = \frac{-2}{10}$



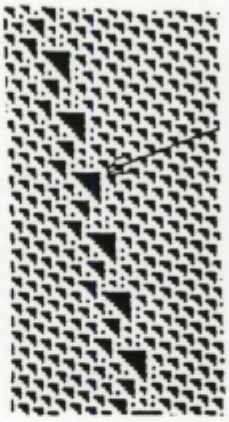
$s = \frac{-2}{10}$

$s = \frac{0}{7}$

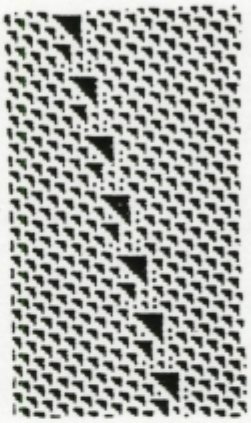
$s = \frac{0}{7}$



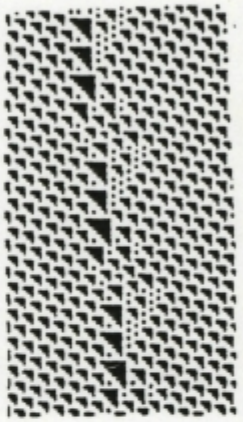
$s = \frac{0}{7}$



$s = \frac{8}{20}$

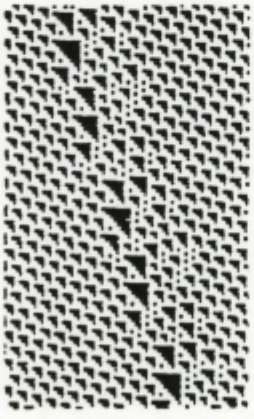


$s = \frac{4}{15}$



$s = \frac{4}{36}$

$s = \frac{8}{30}$

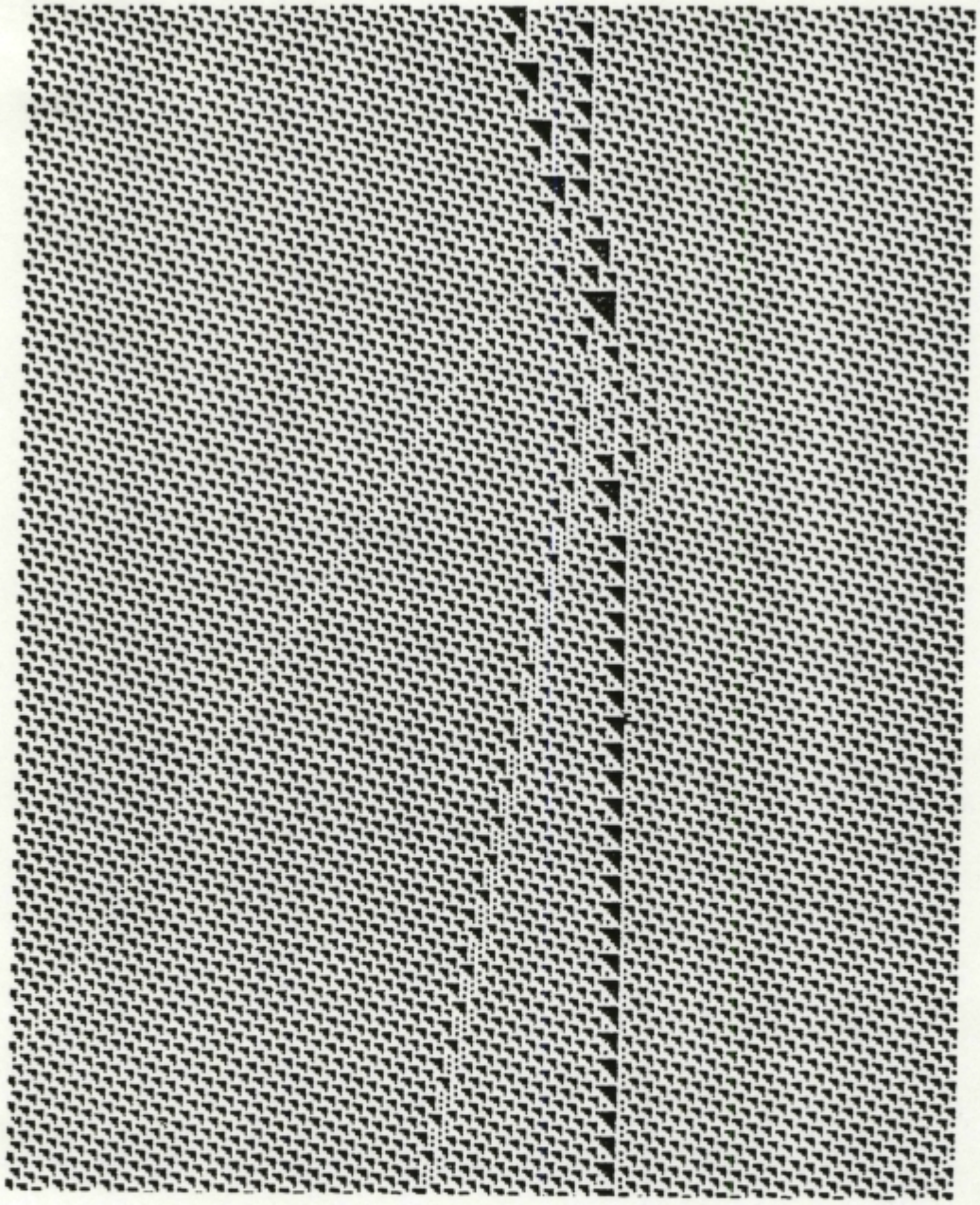


$s = \frac{14}{42}$

$s = \frac{2}{4}$

$s = \frac{6}{12}$

Structures found in the $k=2$, $r=1$ rule with number 193 by Doug Lind, a mathematician at the University of Washington in Seattle. Can you find gliders (or "particles") with other speeds? The values of s give the speeds of those shown here.



Interactions between structures in the $k=2$, $r=1$ rule with number 193. What are all the possible interaction processes? Can one assemble a universal computer using them?