Chapter 25

What is Life?

Life's not always as simple as mathematics, Abraham!
Mrs. Abraham Fraenkel.

Life's too important a matter to be taken seriously.
Oscar Wilde.

... in real life mistakes are likely to be irrevocable. Computer simulation, however, makes it economically practical to make mistakes on purpose. If you are astute, therefore, you can learn much more than they cost. Furthermore, if you are at all discreet, no one but you need ever know you made a mistake.
John McLeod and John Osborn, *Natural Automata and Useful Simulations*, Macmillan, 1966.

Most of this book has been about two-player games, and our last two chapters were about one player games. Now we're going to talk about a no-player game, the **Game of Life**! Our younger readers won't have learned much about Life, so we'd better tell you some of the facts.

Life is a "game" played on an infinite squared board. At any time some of the cells will be **live** and others **dead**. Which cells are live at time 0 is up to you! But then you've nothing else to do, because the state at any later time follows inexorably from the previous one by the rules of the game:

BIRTH. A cell that's *dead* at time t becomes *live* at $t+1$ only if *exactly three* of its eight neighbors were live at t.

DEATH by overcrowding. A cell that's live at t and has four or more of its eight neighbors live at t will be dead by time $t+1$.

DEATH by exposure. A live cell that has only one live neighbor, or none at all, at time t, will also be dead at $t+1$.

These are the only causes of death, so we can take a more positive viewpoint and describe instead the rule for

SURVIVAL. A cell that was live at time t will remain live at $t+1$ if and only if it had just 2 or 3 live neighbors at time t.

Just 3 for BIRTH
2 or 3 for SURVIVAL

A fairly typical Life History is shown in Fig. 1. We chose a simple line of five live cells for our generation 0. In the figures a circle denotes a live cell.

Which of these will survive to generation 1? The two end cells have just one neighbor each and so will die of exposure. But the three inner ones have two living neighbors and so will survive. That's why we've filled in those circles.

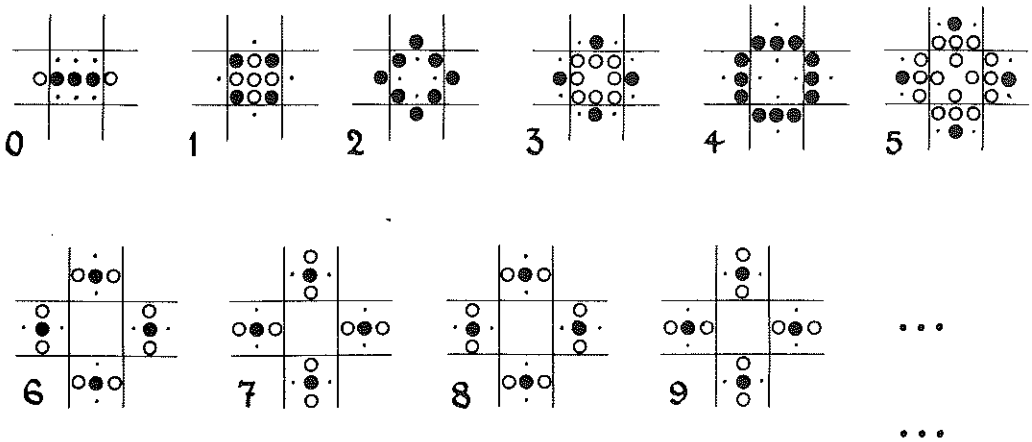


Figure 1. A Line of Five Becomes Traffic Lights.

What about births at time 1? There are three cells on either side of the line that are dead at time 0, but have exactly three live neighbors, so will come to life at time 1. We've shown these prospective births by dots in the figure.

So at time 1 the configuration will be a solid 3×3 square. Let's briefly follow its later progress.

Time 1–2: The corners will survive, having 3 neighbors each, but everything else will die of overcrowding. There will be 4 births, one by the middle of each side.

2–3: We see a ring in which each live cell has 2 neighbors so everything survives; there are 4 births inside.

3–4: Massive overcrowding kills off all except the 4 outer cells, but neighbors of these are born to form:

4–5: another survival ring with 8 happy events about to take place.

5–6: More overcrowding again leaves just 4 survivors. This time the neighboring births form:

6–7: four separated lines of 3, called **Blinkers**, which will never interact again.

7–8–9–10–. . . At each generation the tips of the Blinkers die of exposure but the births on each side reform the line in a perpendicular direction.

The configuration will therefore oscillate with period two forever. The final pair of configurations is sufficiently common to deserve a name. We call them **Traffic Lights**.

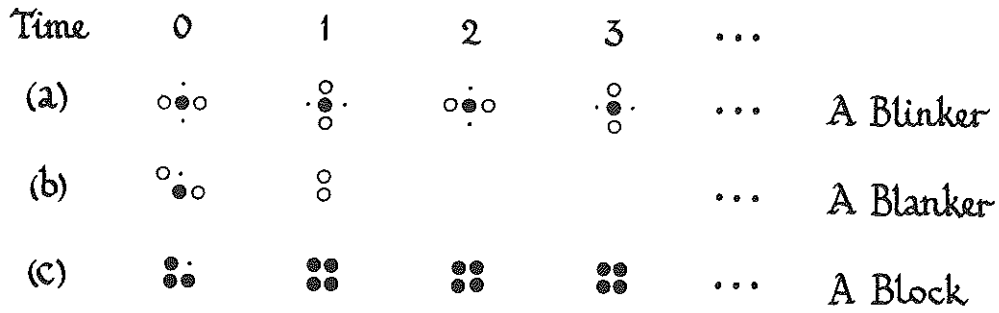


Figure 2. If Three Survive, They'll Make a Blinker or a Block.

The Blinker is also quite common on its own (Fig. 2a). Most other starting configurations of three live cells will blank out completely in two moves (Fig. 2(b)). But if you start with three of the four cells of a 2×2 block, the fourth cell will be born and then the **Block** will be stable (Fig. 2(c)) because each cell is neighbored by the three others.

STILL LIFE

It's easy to find other stable configurations. The commonest such **Still Life** can be seen in Fig. 3 along with their traditional names. The simple cases are usually loops in which each live cell has two or three neighbors according to local curvature, but the exact shape of the loop is important for effective birth control.

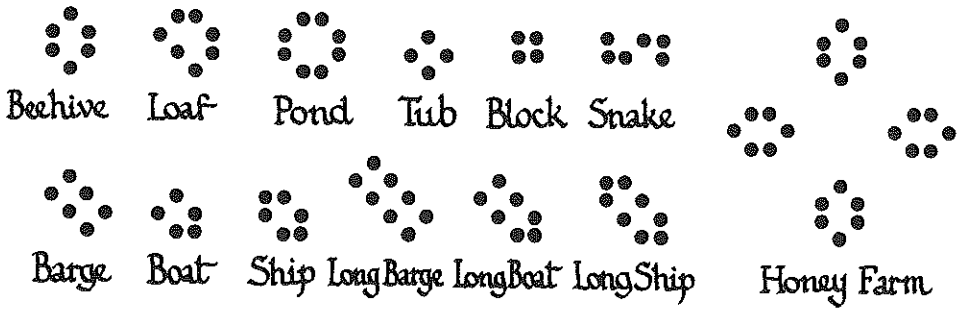


Figure 3. Some of the Commoner Forms of Still Life.

LIFE CYCLES

The blinker is the simplest example of a configuration whose life history repeats itself with period >1. Lifenthusiasts (a word due to Robert T. Wainwright) have found many other such configurations, a number of which are shown in Figs. 4 to 8.

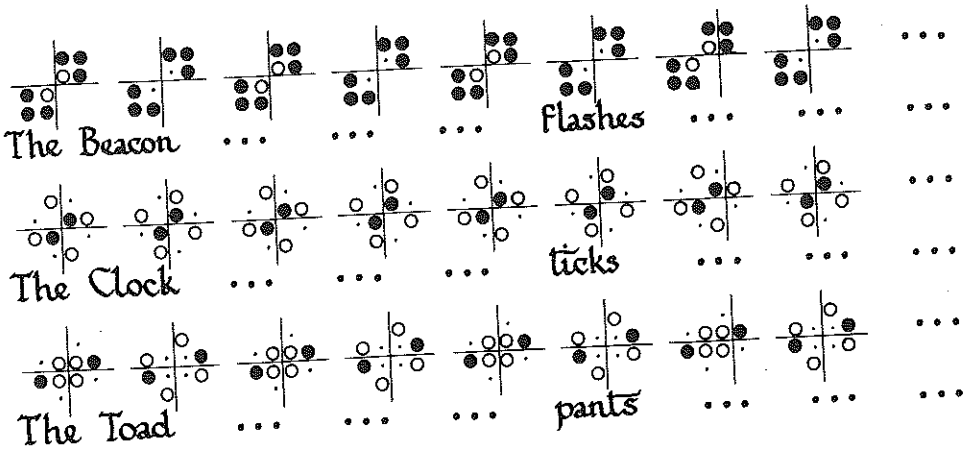


Figure 4. Three Life Cycles with Period Two.

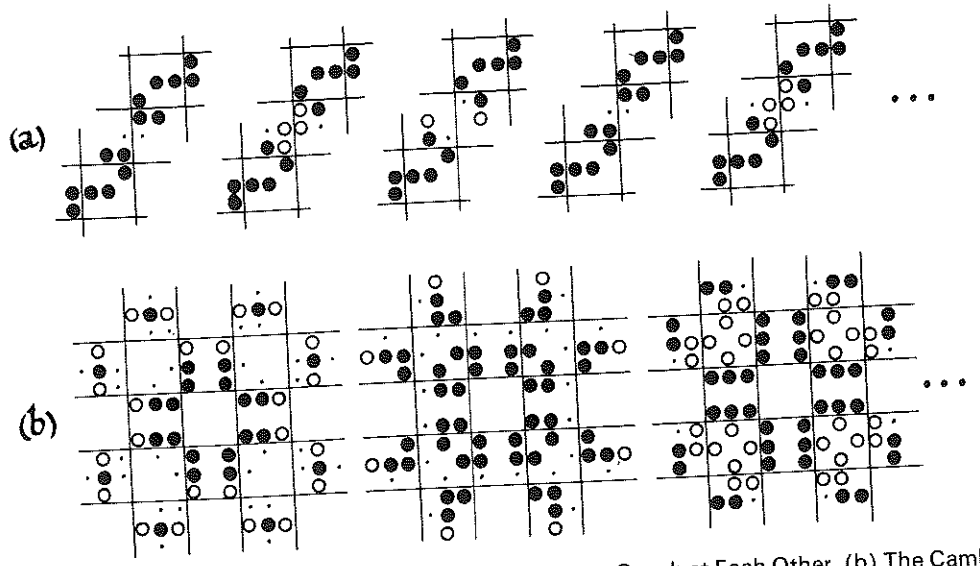


Figure 5. Two Life Cycles with Period Three. (a) Two Eaters Gnash at Each Other. (b) The Cambridge Pulsar CP 48-56-72.

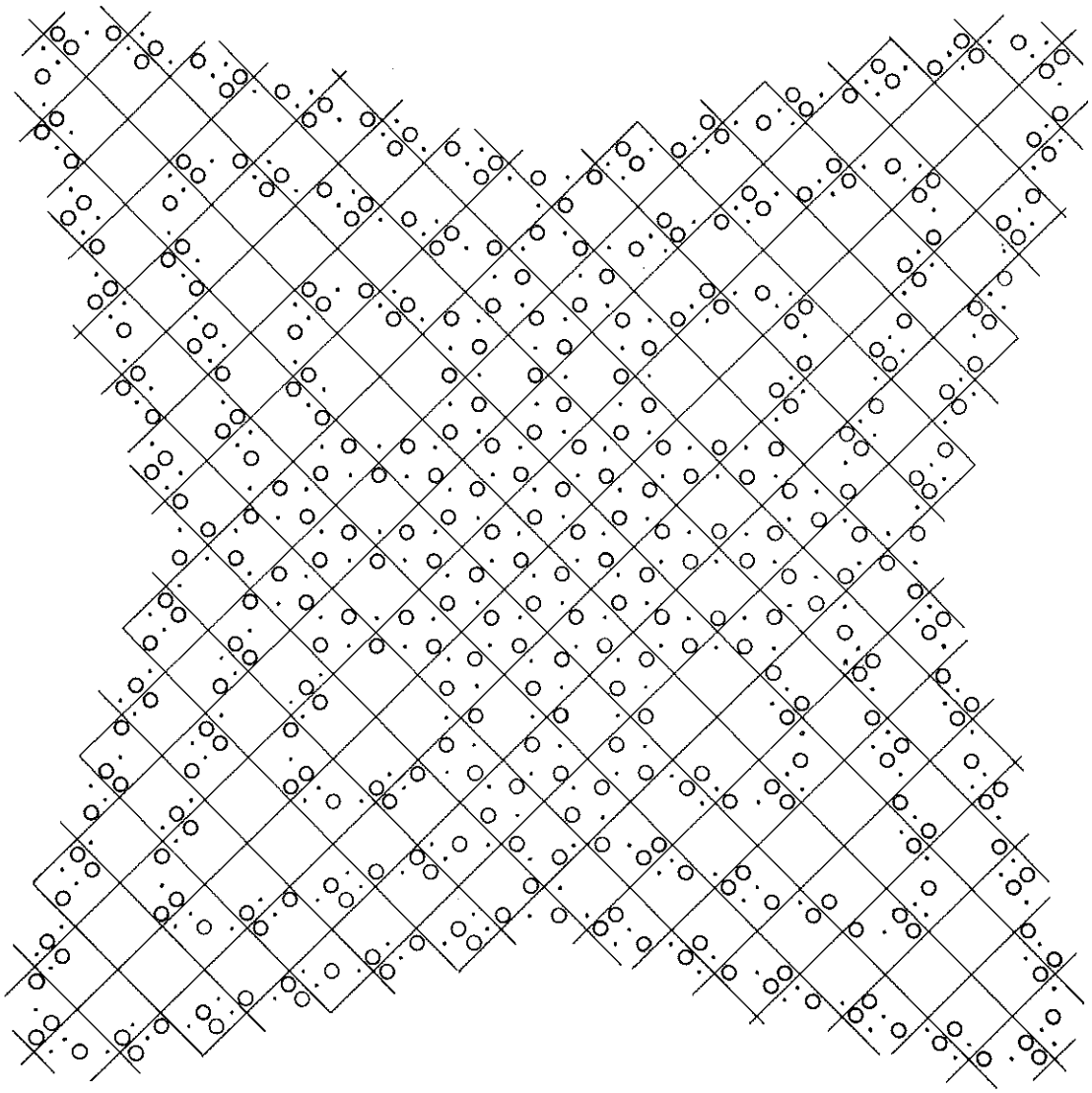


Figure 6. A Flip-Flop by the Gosper Group.

THE GLIDER AND OTHER SPACE SHIPS

When we first tracked the *r*-pentomino (you'll hear about that soon) some guy suddenly said, "Come over here, there's a piece that's walking!" We came over and found Fig. 9.

You'll see that generation 4 is just like generation 0 but moved one diagonal place, so that the configuration will steadily move across the plane. Because the arrangements at times 2, 6, 10, ... are related to those at times 0, 4, 8, 12, ... by the symmetry that geometers call a *glide reflexion*, we call this creature the **glider**. But when you see Life played at the right speed by a computer

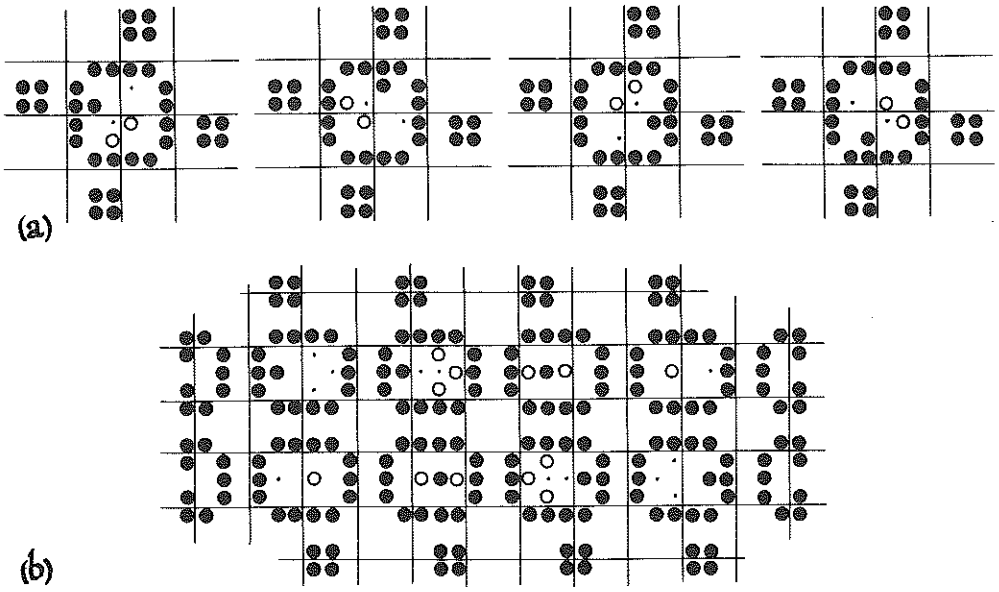


Figure 7. (a) Catherine Wheel. (b) Hertz Oscillator. Still Life Induction Coils Keep Field Stable.

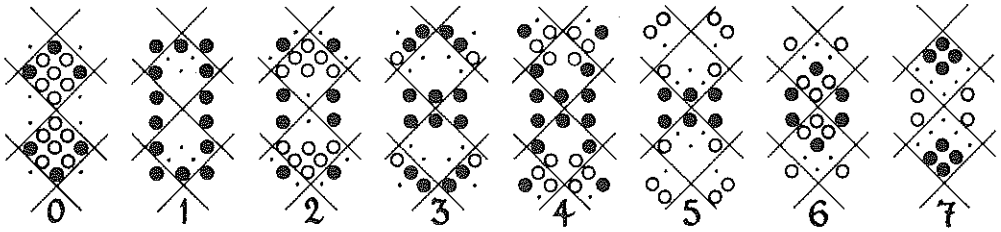


Figure 8. Figure 8.

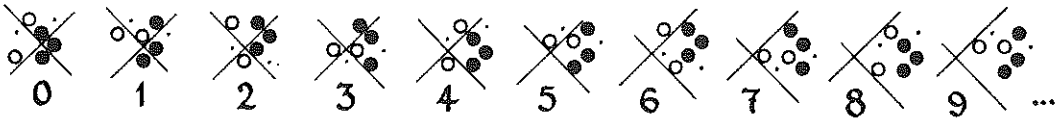


Figure 9. The Glider Moves One Square Diagonally Each Four Generations.

on a visual display, you'll see that the glider walks quite seductively, wagging its tail behind it. We'll see quite a lot of the glider in this chapter.

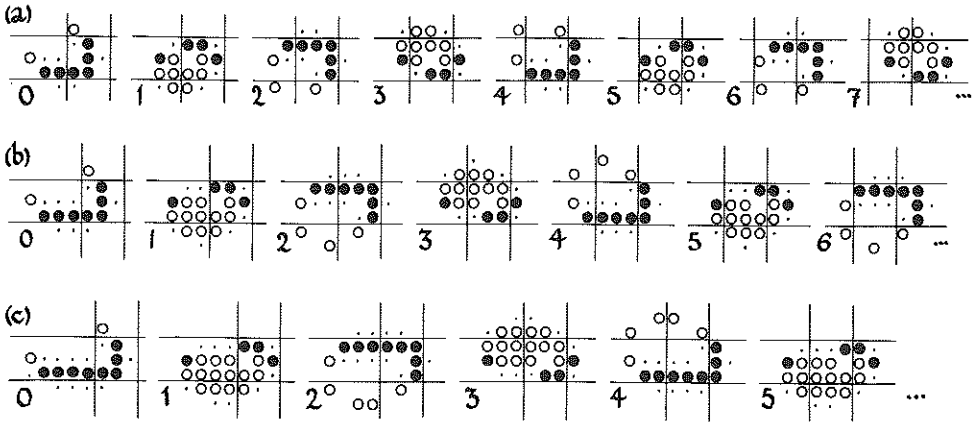


Figure 10. (a) Lightweight, (b) Middleweight, (c) Heavyweight Spaceships.

It was at just such a visual computer display that one of us first noticed the **spaceship** of Fig. 10(a) (and was very lucky to be able to stop the machine just before it would have crashed into another configuration). This **lightweight spaceship** immediately generalizes to the **middleweight** and **heavyweight** ones (Figs. 10(b) and (c)) but longer versions turn out to be unstable. It was later discovered, however, that arbitrarily long spaceships can still travel provided they are suitably escorted by small ones (Fig. 11). All the spaceships, as drawn, move Eastwards.

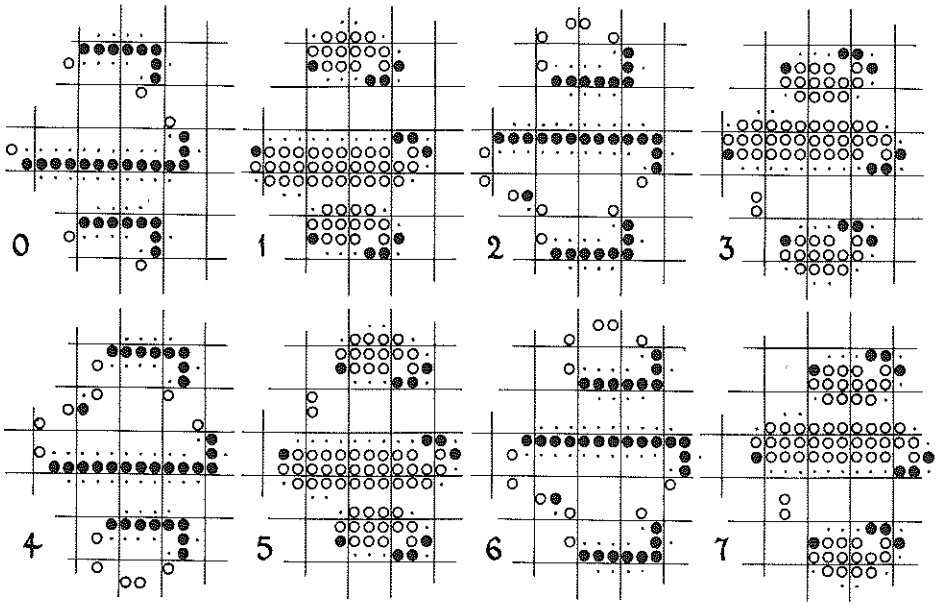


Figure 11. An Overweight Spaceship Escorted by Two Heavyweight Ones.

THE UNPREDICTABILITY OF LIFE

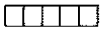
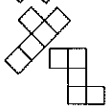
Is there some way to foretell the destiny of a Life pattern? Will it eventually fade away completely? Or become static? Oscillate? Travel across the plane, or maybe expand indefinitely?

Let's look at what *should* be a very simple starting configuration—a straight line of n live cells.

$n = 1$ or 2	fades immediately,
$n = 3$	is the Blinker;
$n = 4$	becomes a Beehive at time 2,
$n = 5$	gave Traffic Lights (Fig. 1) at time 6,
$n = 6$	fades at $t = 12$,
$n = 7$	makes a beautifully symmetric display before terminating in the **Honey Farm** (Fig. 3) at $t = 14$;
$n = 8$	gives 4 blocks and 4 beehives,
$n = 9$	makes two sets of Traffic Lights,
$n = 10$	turns into the **pentadecathlon**, with a life cycle of 15,
$n = 11$	becomes two blinkers,
$n = 12$	makes two beehives,
$n = 13$	turns into two blinkers,
$n = 14$ and $n = 15$	vanish completely,
$n = 16$	makes a big set of Traffic Lights with 8 blinkers,
$n = 17$	becomes 4 blocks
$n = 18$ and $n = 19$	fade away entirely,
$n = 20$	makes just 2 blocks,

and so on.

What's the general pattern? Even when we follow the configurations which start with a very small number of cells, it's not easy to see what goes on. There are 12 edge-connected regions with 5 cells (S.W. Golomb calls them pentominoes). Here are their histories:

Figure	Our mnemonic	Destiny
		becomes Traffic Lights (Fig. 1) at time 6.
		fades at time 4.
		becomes Traffic Lights at time 9.
		reaches a steady state at time 1103, consisting of 4 Blinkers, 1 Ship, 1 Boat, 1 Loaf, 4 Beehives, and 8 Blocks, having emitted 6 Gliders! (For a picture, see p. 105 of the Sci. Amer. for Jan. 1971.)
		fades at time 4.
		becomes Traffic Lights at time 10.
		fades at time 4.
		becomes W at time 1, then
		becomes a Loaf 2 generations later.
		becomes Traffic Lights at time 6.
		fades at time 3, and so does

Once again, it doesn't seem easy to detect any general rule.

Here, in Figs. 12 and 13 are some other configurations with specially interesting Life Histories, for you to try your skill with.

Can the population of a Life configuration grow without limit? Yes! The $50.00 prize that one of us offered for settling this question was won in November 1970 by a group at M.I.T. headed by R.W. Gosper. Gosper's ingenious **glider gun** (Fig. 14) emits a new glider every 30 generations. Fortunately it was just what we wanted to complete our proof that

Life is really unpredictable!

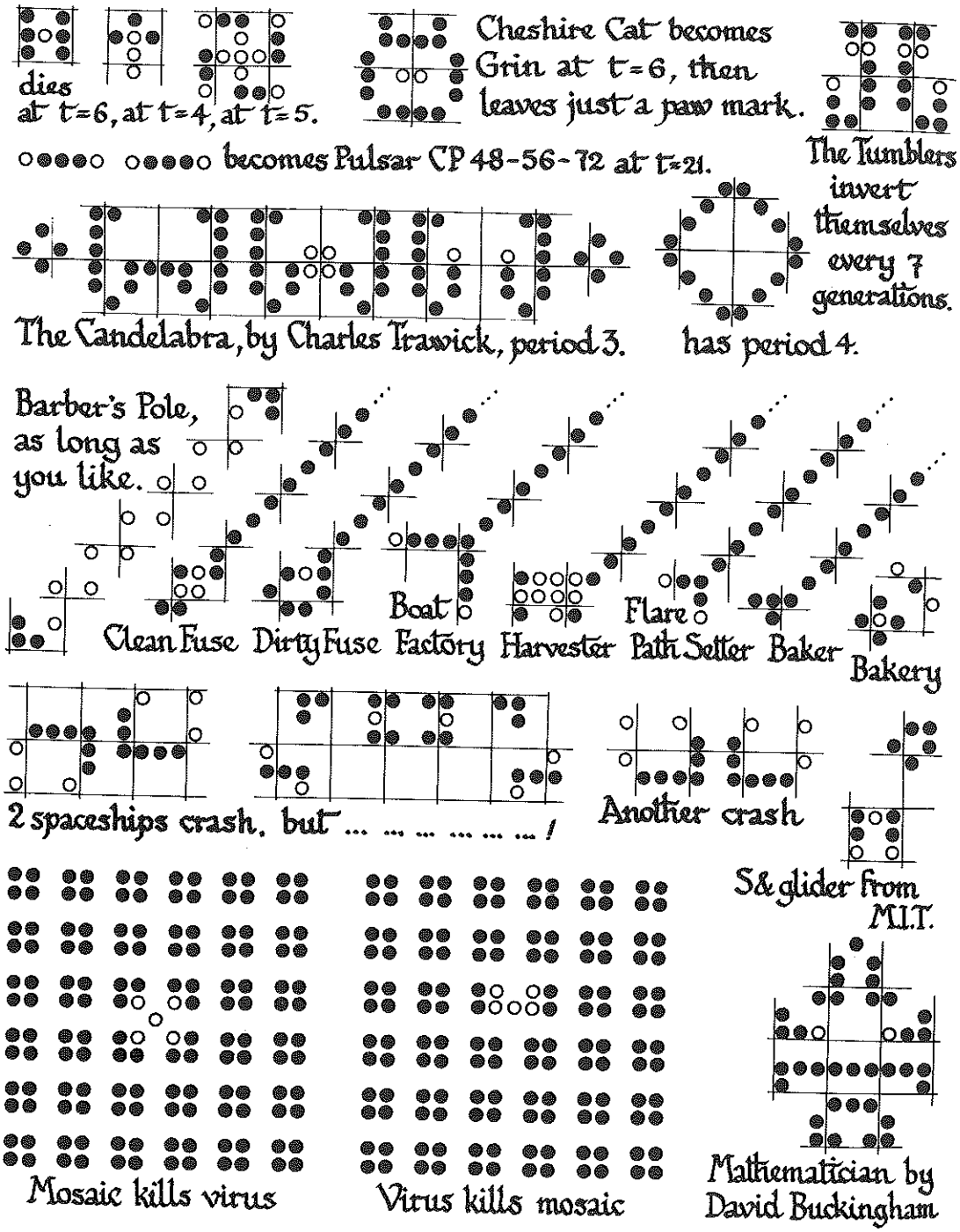


Figure 12. Exercises for the Reader.

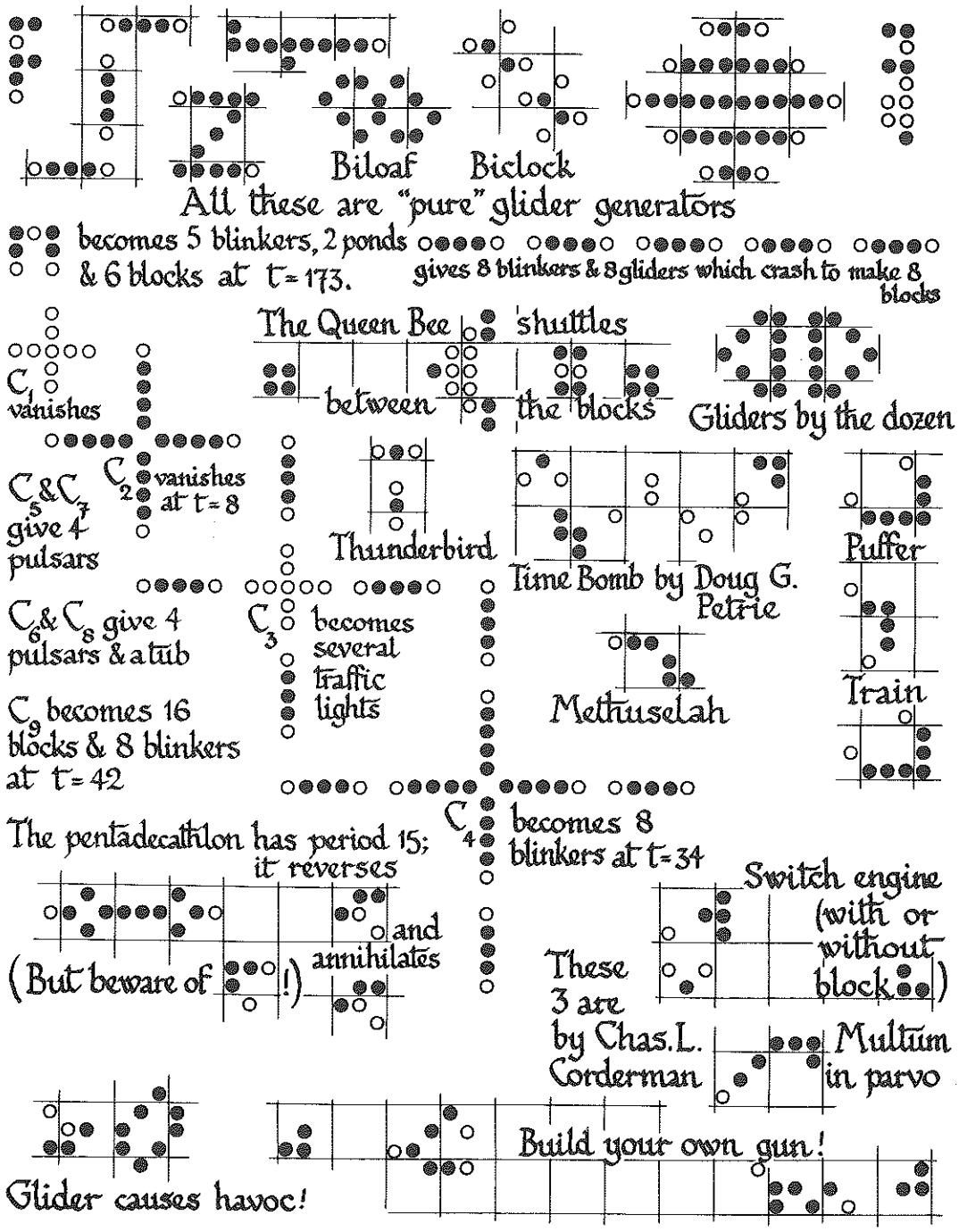


Figure 13. Mainly for Computer Buffs.

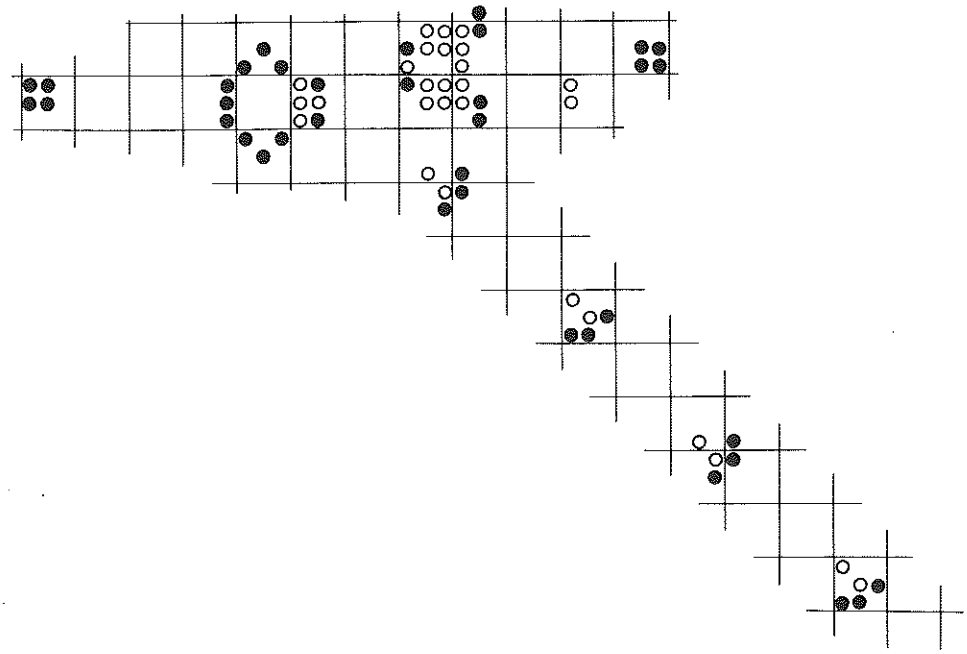


Figure 14. Gosper's Glider Gun.

GARDENS OF EDEN

There are Life configurations that can only arise as the initial state, because they have no ancestors!

We'll prove that if n is sufficiently large, there is some configuration within a $5n-2$ by $5n-2$ square that has no parent. It will suffice to examine that part of a prospective parent that lies in the surrounding $5n \times 5n$ square (Fig. 15). If any one of the component 5×5 squares is empty, it can be replaced as in Fig. 15(b) without affecting subsequent generations. So we need consider only

$$(2^{25}-1)^{n^2} = 2^{24\cdot 999999957004337\ldots n^2} \text{ of the } 2^{25n^2}$$

configurations in the $5n$ by $5n$ square. However, there are exactly

$$2^{(5n-2)^2} = 2^{25n^2-20n+4}$$

possible configurations in the $5n-2$ by $5n-2$ square, so that if

$$24\cdot 999999957004337\ldots n^2 < 25n^2 - 20n + 4,$$

one of these will have no parent! We calculate that this happens for $n = 465163200$, so that there is a Garden of Eden configuration that will fit comfortably inside a

2325816000 by 2325816000 square!

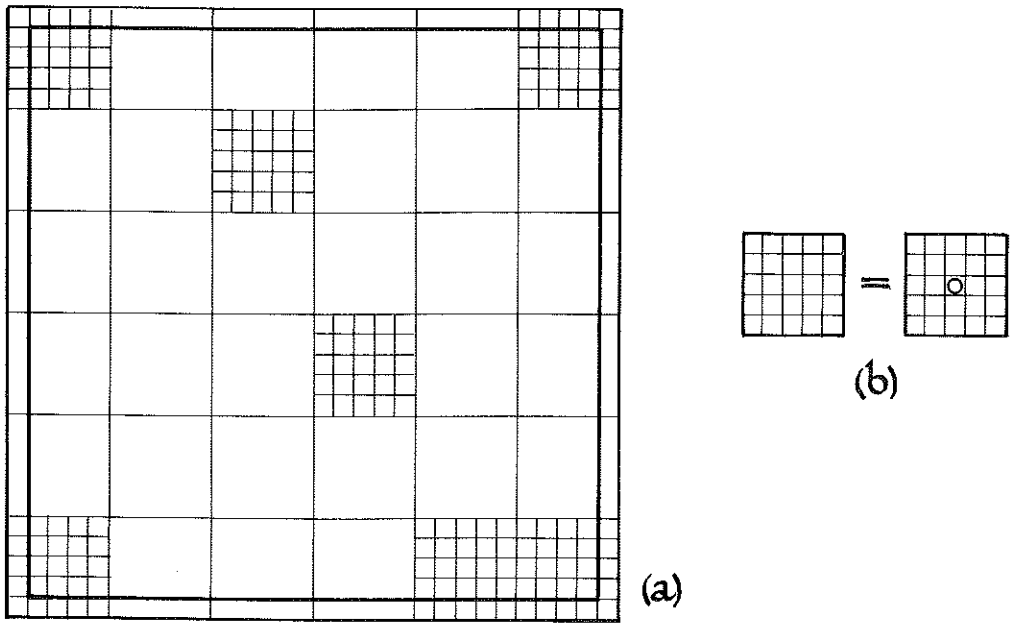


Figure 15. Location of the Garden of Eden.

This type of argument was first used by E.F. Moore in a more general context. More careful counting in the Life case has brought the size down to 1400 by 1400. However, using completely different ideas and many hours of computer time the M.I.T. group managed to produce an explicit example (Fig. 16).

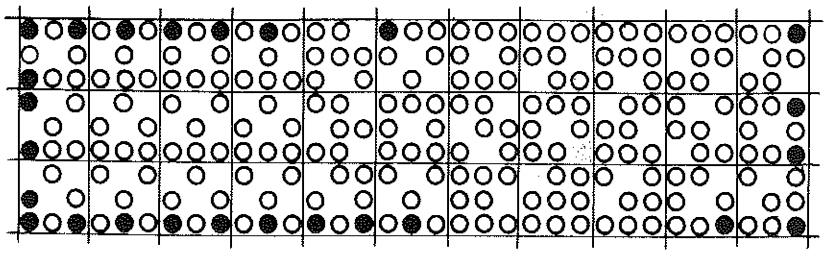


Figure 16. An Orphan Found by Roger Banks, Mike Beeler, Rick Schroeppel, Steve Ward, *et al.*

LIFE'S PROBLEMS ARE HARD!

The questions we posed about the ultimate destiny of Life configurations may not seem very mathematical. After all, Life's but a game! Surely there aren't any difficult mathematical problems there?

Well, yes there are! Indeed we can prove the astonishing fact that *every* sufficiently well-stated mathematical problem can be reduced to a question about Life! Those apparently trivial problems about Life histories can be arbitrarily difficult!

Here, for instance, is a tricky little problem that's kept mathematicians busy ever since Pierre de Fermat proposed it over 300 years ago. Is it possible for a perfect nth power to be the sum of two smaller ones for any n larger than 2? Despite many learned investigations by many learned mathematicians we still don't know! But if you had an infallible way to foretell the destiny of a given Life configuration, you'd be able to answer this question!

The reason is that we can design for you a finite starting pattern P_0 which will fade away completely if and only if there is a way of breaking an nth power into two smaller ones. If you had a mechanical method which would accept as input an arbitrary finite Life pattern P, and is guaranteed to respond with

> FADE, if the rules of Life will eventually cause P to disappear completely, and
> STAY, if not,

then you could apply it to P_0 and settle Fermat's question.

Even better, we could design a pattern P_1 which will tell you what those perfect powers are. If

$$a^n + b^n = c^n$$

☞ is the first solution of Fermat's problem in a certain dictionary order, then eventually P_1 will lead to a configuration in which there are

> a gliders, travelling North–West,
> b gliders, travelling North–East,
> c gliders, travelling South–West,
> n gliders, travelling South–East,

and nothing else at all! We can do the same sort of thing for other mathematical problems.

MAKING A LIFE COMPUTER

Many computers have been programmed to play the game of Life. We shall now return the compliment by showing how to define Life patterns that can imitate computers. Many remarkable consequences will follow from this idea.

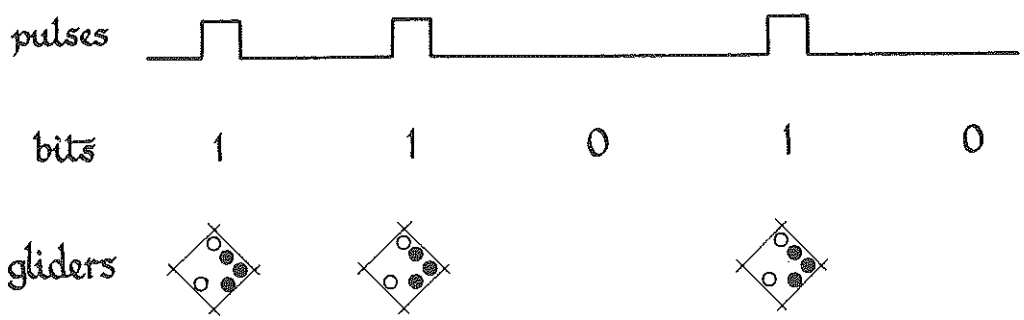


Figure 17. Gliding Pulses.

Good old fashioned computers are made from pieces of wire along which pulses of electricity go. Our basic idea is to mimic these by certain lines in the plane along which gliders travel (Fig. 17). (Because gliders travel diagonally, from now on we'll turn the plane through 45°, so they move across, or up and down, the page.) Somewhere in the machine there is a part called the **clock** which generates pulses at regular intervals and most of the working parts of the machine are made up of logical **gates**, like those drawn in Fig. 18. Obviously we can use Glider Guns as pulse generators. What should we do about the logical gates? Let's study the possible interactions of two gliders which crash at right angles.

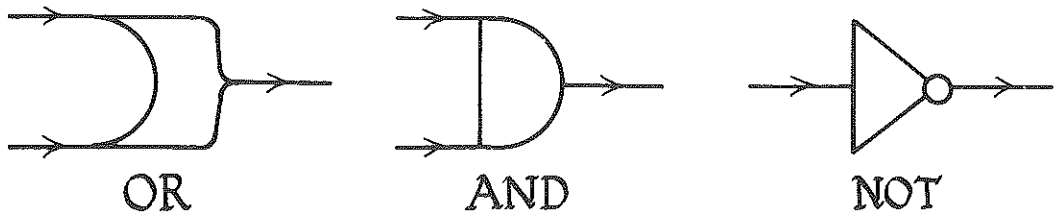


Figure 18. The Three Logical Gates.

WHEN GLIDER MEETS GLIDER

There are lots of different ways in which two gliders can meet, because there are lots of different possibilities for their exact arrangement and timing. Figure 19 shows them crashing (a) to form a blinker, (b) to form a block, (c) to form a pond, or (d) in one of several ways in which they can annihilate themselves completely. This last may seem rather unconstructive, but these **vanishing reactions** turn out to be surprisingly useful!

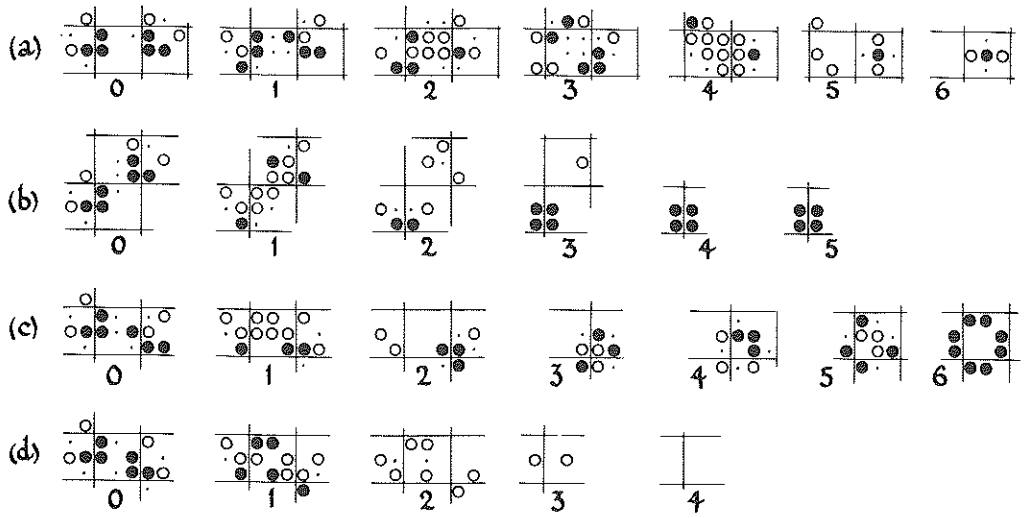


Figure 19. Gliders Crashin' in Diverse Fashion.

HOW TO MAKE A NOT GATE

We can use a vanishing reaction, together with a Glider Gun, to create a NOT gate (Fig. 20). The input stream enters at the left of the figure, and the Glider Gun is positioned and timed so that every space in the input stream allows just one glider to escape from the gun, while a glider in the stream necessarily crashes with one from the gun in a vanishing reaction (indicated by ∗). Figure 20 shows the periodic stream

1 1 0 1 1 0 1 1 0 ...

being complemented to

0 0 1 0 0 1 0 0 1 ....

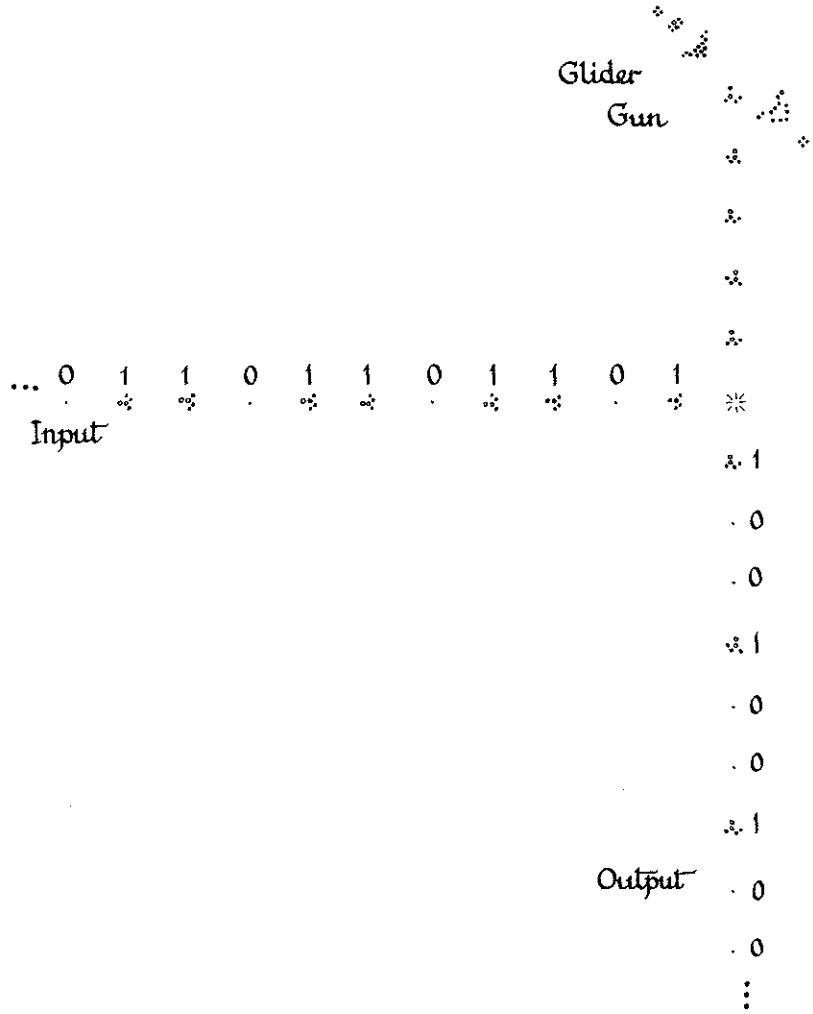


Figure 20. A Glider Gun and a Vanishing Reaction Make a NOT Gate.

Fortunately there are several vanishing reactions with different positions and timings in which the decay is so fast that later gliders from the same gun stream will not be affected (Fig. 21). This means that we can reposition a glider stream arbitrarily by turning it through sufficiently many corners (Fig. 22).

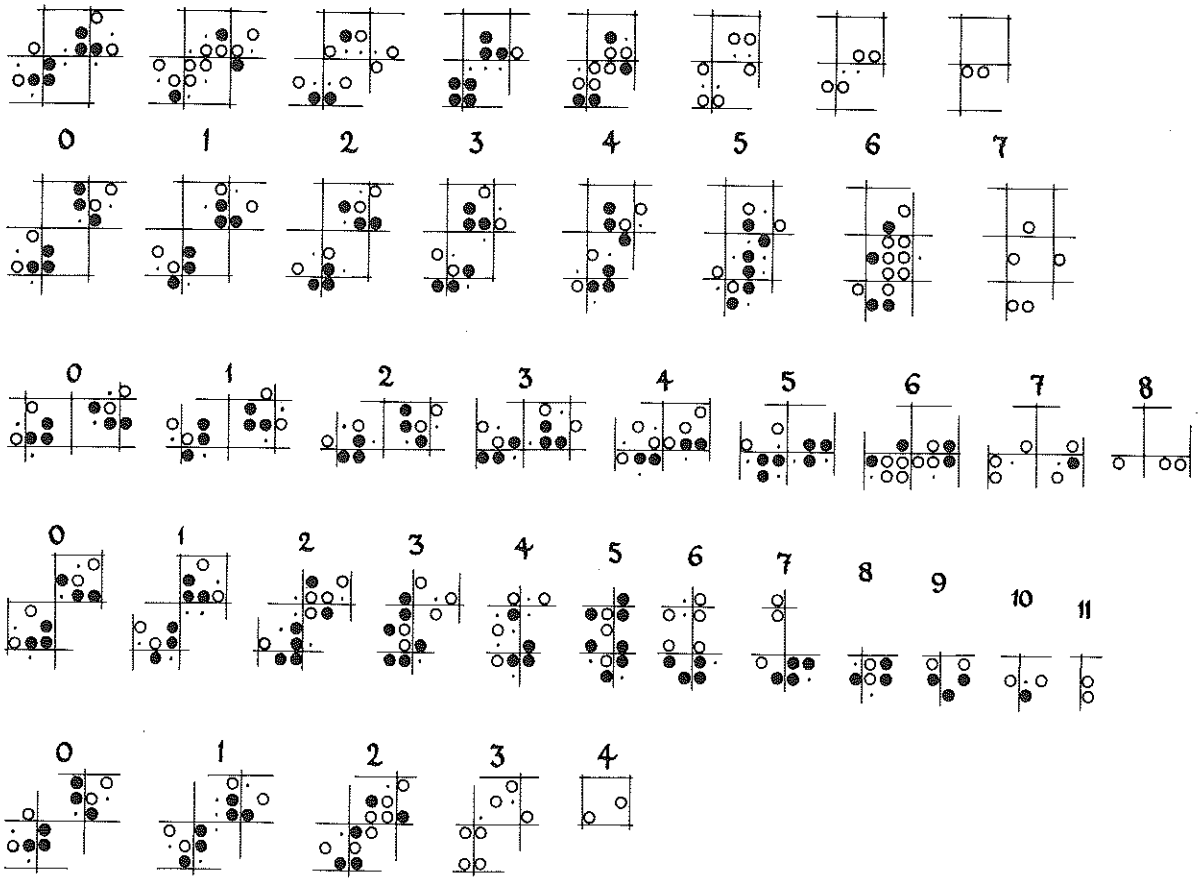


Figure 21. A Variety of Vanishing Reactions Between Crashing Gliders.

THE EATER

What else can happen when glider meets glider? Lots of things! One of them is to make an **eater** (Fig. 23) and an eater can eat lots of things without suffering any indisposition. The eater, which was discovered by Gosper, will be very useful to us; in Fig. 24 you can see it enjoying a varied diet of (a) a blinker, (b) a pre-beehive, (c) a lightweight spaceship, (d) a middleweight spaceship, and (e) a glider. If it attempts a heavyweight spaceship it gets indigestion and leaves a loaf behind; if it tries a blinker in the wrong orientation it leaves a baker's shop!

Sometimes glider streams are embarrassing to have around, so it's especially useful then—it just sits there and eats up the whole stream!

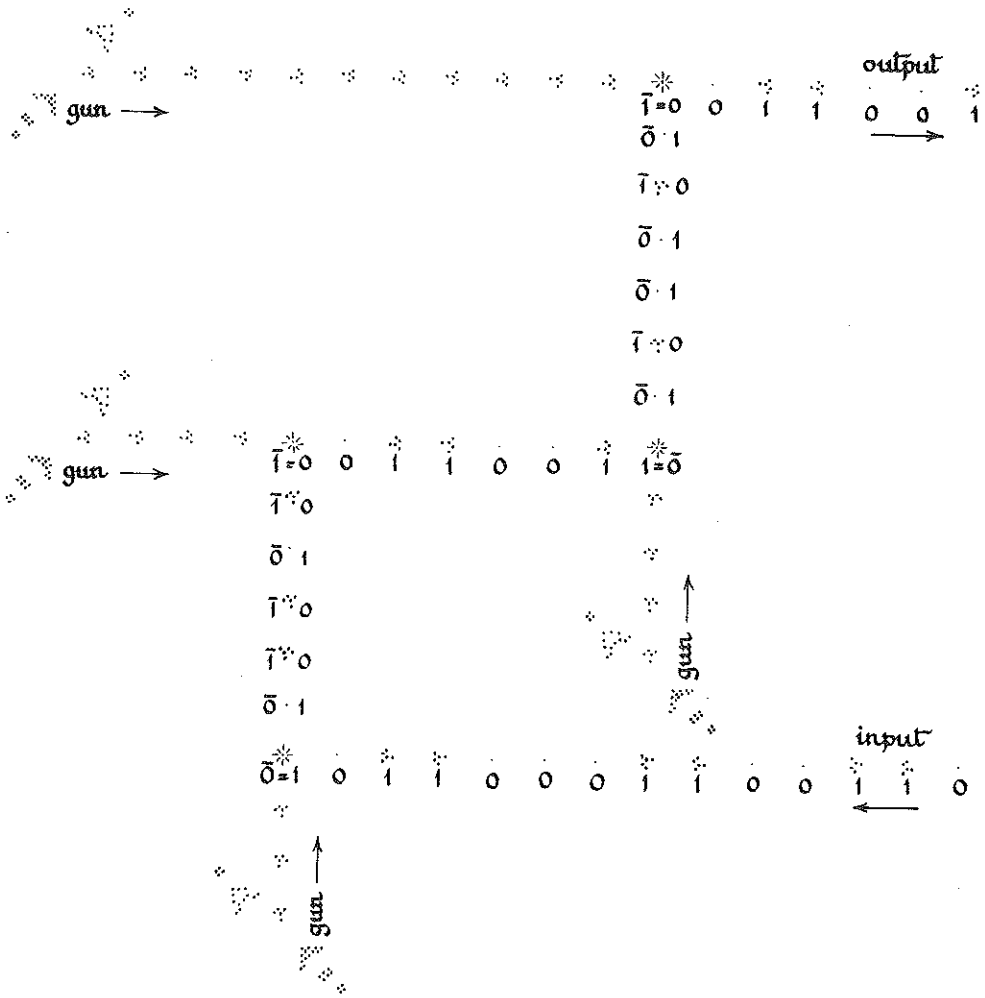


Figure 22. Repositioning and Delaying a Glider Stream.

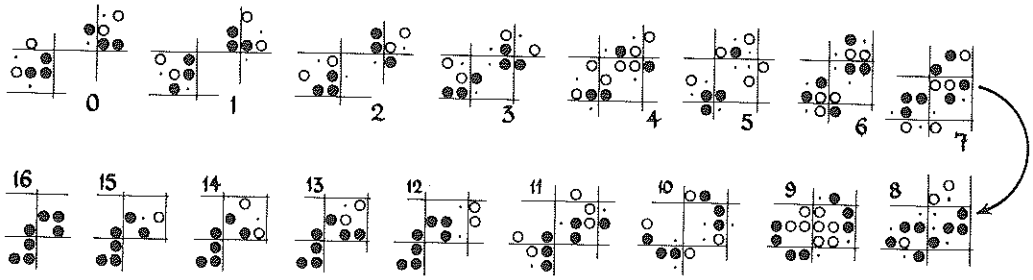


Figure 23. Two Gliders Crash to Form an Eater.

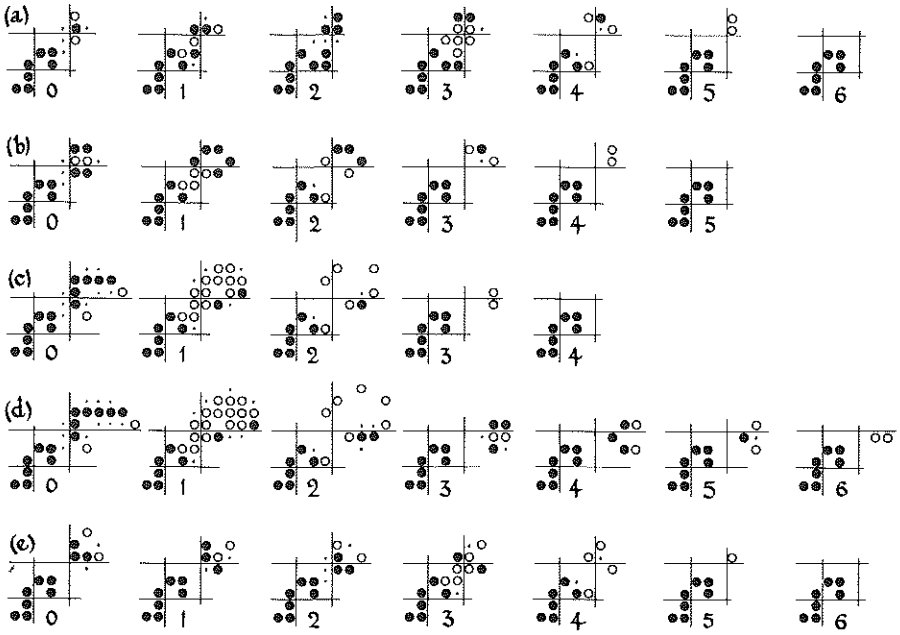


Figure 24. The Voracious Eater Devours a Varied Meal.

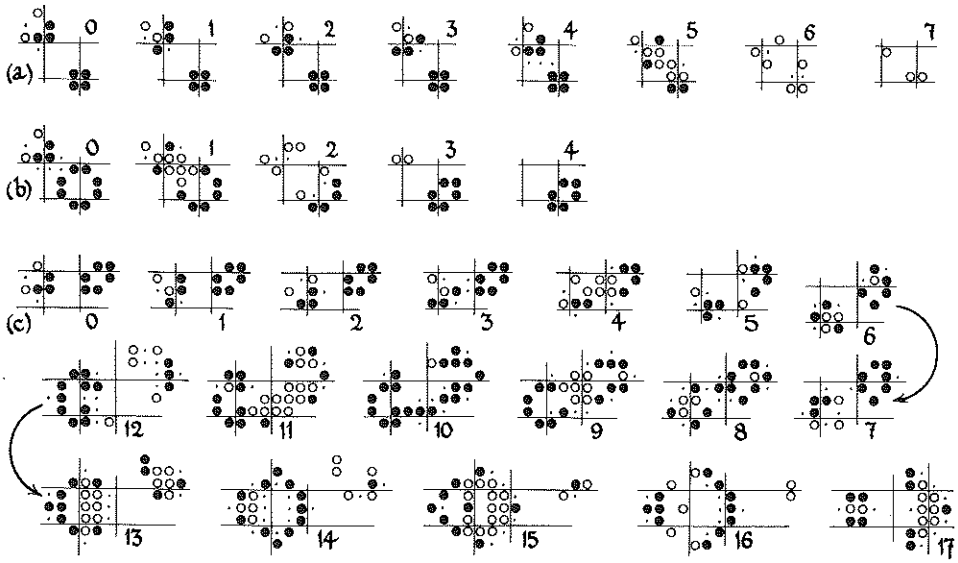


Figure 25. (a) Blockbusting Glider. (b) Glider Dives into Pond and Comes Up With Ship. (c) Glider Crashes into Ship and Makes Part of Glider Gun.

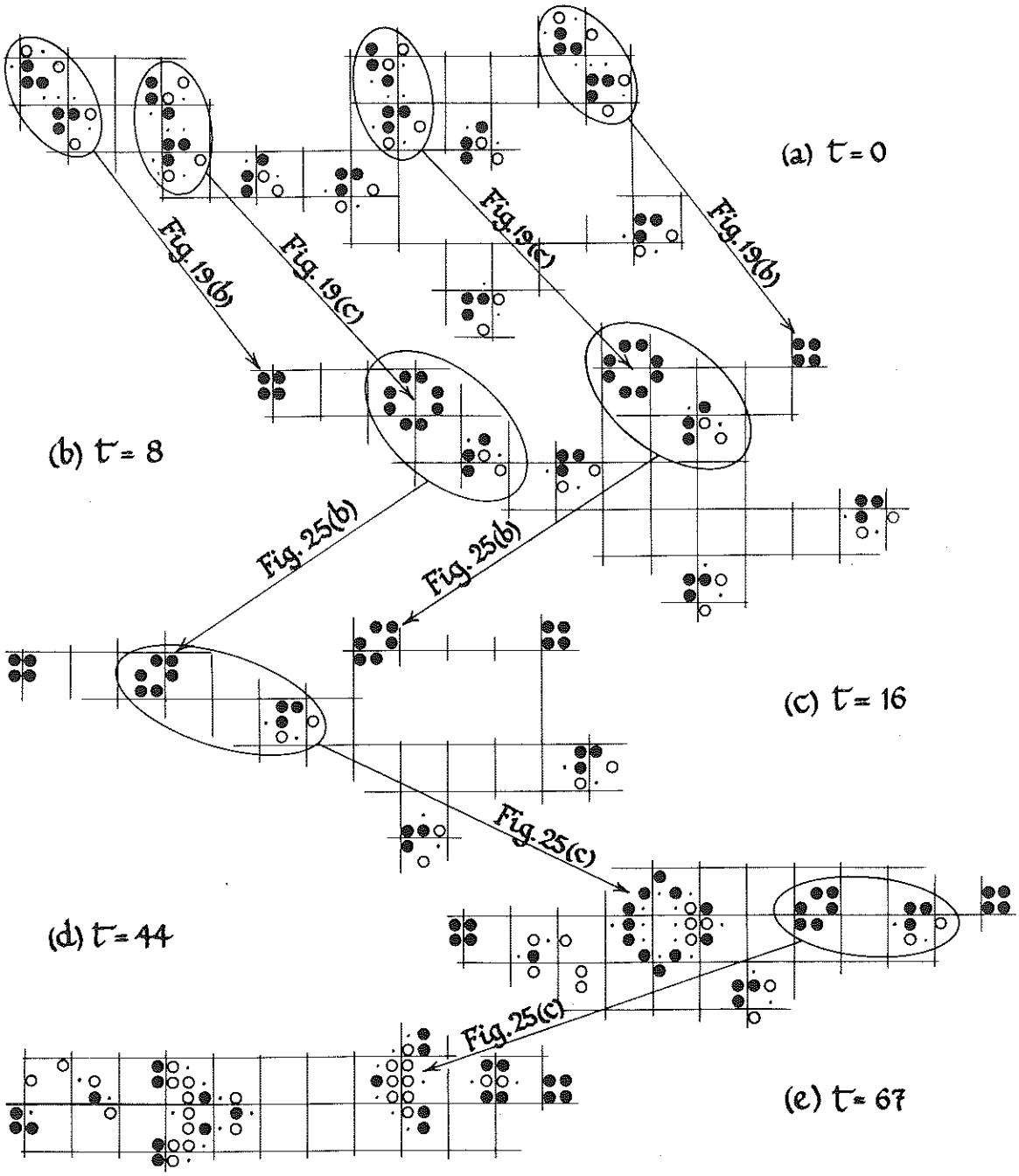


Figure 26. Thirteen Gliders Build Their Own Gun.

GLIDERS CAN BUILD THEIR OWN GUNS!

What happens when a glider meets other things? We have seen it get eaten by an eater. It can also annihilate a block (and itself! Fig. 25(a)). But more constructively it can turn a pond into a ship (Fig. 25(b)) and a ship into a part of the glider gun (Fig. 25(c)). And since gliders can crash to make blocks (Fig. 19(b)) and ponds (Fig. 19(c)), they can make a whole glider gun! The 13 gliders in Fig. 26(a) do this in 67 generations. Figures 26(b,c,d) show the positions after 8, 16 and 44 generations. The extra glider then slips in to deal with an incipient beehive, and by 67 generations (Fig. 26(e)) the gun is in full working order and launches its first glider 25 generations later.

THE KICKBACK REACTION

Yet another very useful reaction between gliders is the **kickback** (Fig. 27(a)) in which the decay product is a glider travelling along a line closely parallel to one of the original ones, but in the opposite direction. We think of this glider as having been *kicked back* by the other one. Figure 27(b) shows our notation for the kickback.

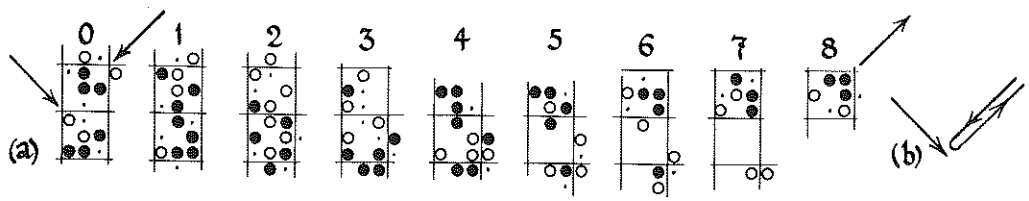


Figure 27. The Kickback.

All the working parts of our computer will be moving glider streams, meeting in vanishing and kickback reactions. The only static parts will be glider guns and eaters (indicated by G and E in the figures).

THINNING A GLIDER STREAM

The glider streams that emerge from normal guns are so dense that they cannot interpenetrate without interfering. If we try to build a computer using streams of this density we couldn't allow any two wires of this kind to cross each other, so we'd better find some way to reduce the pulse rate.

In Fig. 28 the guns G_1 and G_2 produce normal glider streams in parallel but opposite directions. But there is a glider g which will travel West until at A it is kicked East by a glider from the G_1 stream. The timing and phasing are such that at B it will be kicked back towards A again, so that it repeatedly "loops the loop", removing one glider from each of the two streams per cycle. After this every Nth glider is missing from each of these streams. We don't want the G_1 stream, so we feed it into an eater, but we feed the G_2 stream into a vanishing reaction with a stream from a third gun G_3. Every glider from G_2 now dies, but every Nth one from G_3 escapes through a hole in the G_2 stream! So the whole pattern acts as a **thin gun**, producing just one Nth as many gliders as the normal gun. To get the phasing right, N must be divisible by 4, but it can be arbitrarily

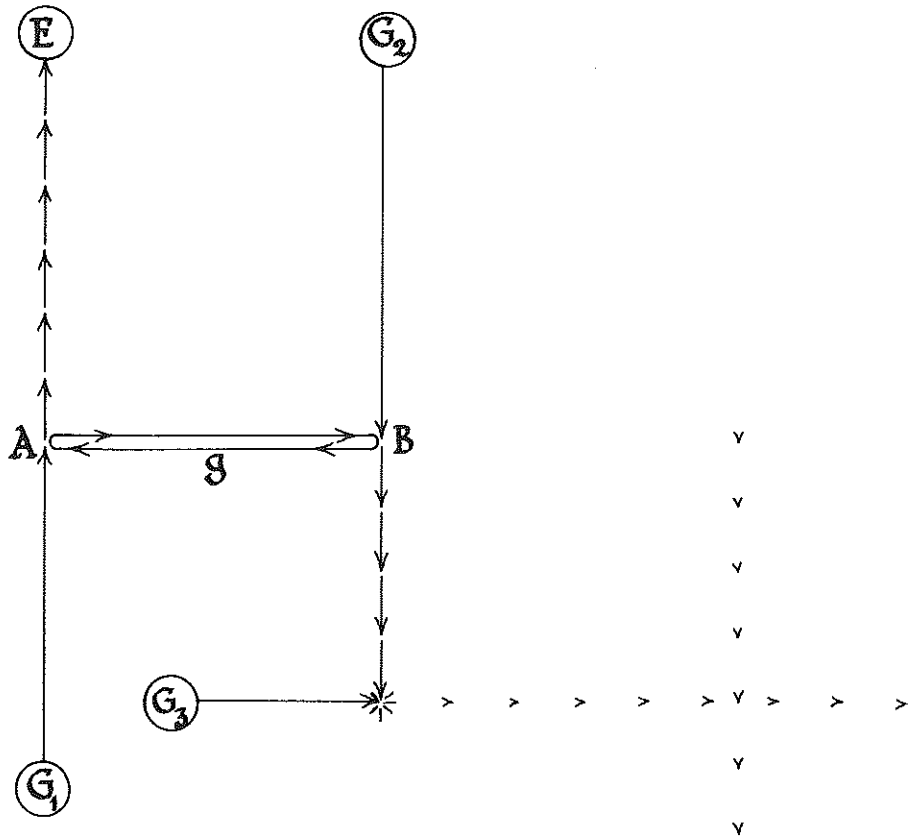


Figure 28. Thinning a Glider Stream.

large and so we can make an arbitrarily thin stream. Now two such streams can cross without interacting as in the right hand part of Fig. 28, provided things are properly timed. So from now on we can use the word **gun** to mean an arbitrarily thin gun. Perhaps a thinning factor of 1000 will make all our constructions work.

BUILDING BLOCKS FOR OUR COMPUTER

In Fig. 29 we see how to build logical gates using only vanishing reactions (we've already seen the NOT gate in more detail in Fig. 20). But there's a problem! The output streams from the AND and OR gates are *parallel* to the input, but the output stream from the NOT gate is at *right angles* to the input. We need a way to turn streams round corners without complementing them, or of complementing them without turning them round corners. Fortunately the solution to our *next* problem automatically solves *this* one.

The new problem is to provide several copies of the information from a given glider stream, and we found it a hard problem to solve. To get some clues, let's see what happens when we use one glider to kick back a glider from a gun stream.

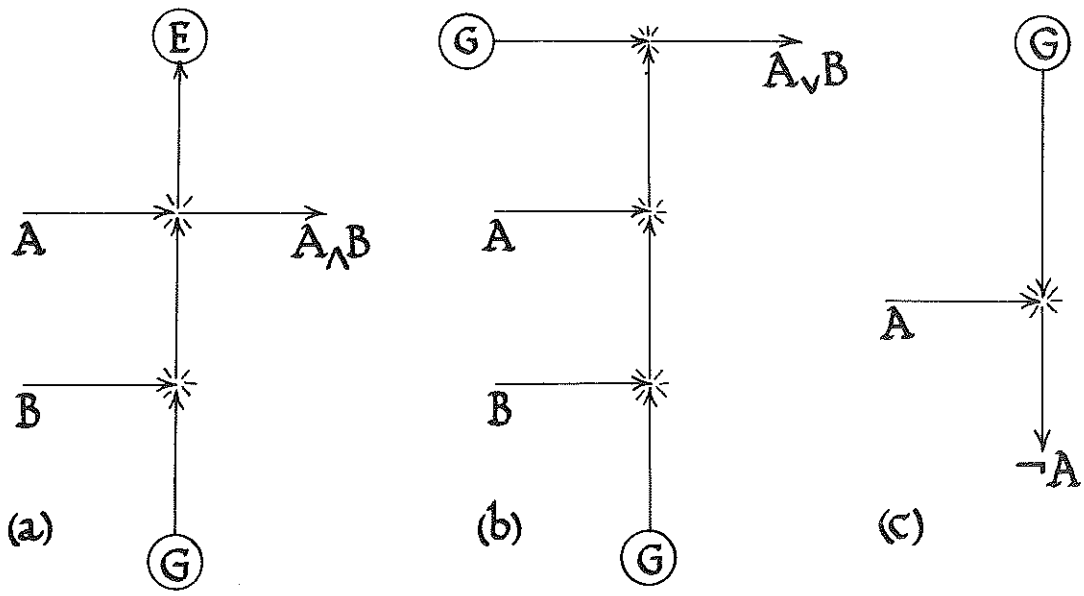


Figure 29. (a) An AND Gate. (b) An OR Gate. (c) A NOT Gate.

We suppose that the gun stream, the **full stream**, produces a glider every 120 generations (a quarter of the original gun density; $N=4$ in the previous section). Then it turns out that when we kick back the first glider, the effect is to remove just three gliders from the stream! This happens as follows:

(i) The first glider is kicked back (Fig. 27) along the full stream.
(ii) The second glider crashes into the first, forming a block (somewhat as in Fig. 19(b)).
(iii) The third glider annihilates the block (Fig. 25(a)).
(iv) All subsequent gliders from the full stream escape unharmed.

We can use this curious behavior as follows. Suppose that our information-carrying stream operates at one tenth, say, of the density of the full stream, so that the last 9 of every 10 places on it will be empty, while the first place might or might not be full. If we use 0 for a hole and block the places in tens, our stream looks like

$$\ldots\ 000000000D\ \ 000000000C\ \ 000000000B\ \ 000000000A \rightarrow$$

We first feed it into an OR gate with a stream of type

$$\ldots\ 00000000g0\ \ 00000000g0\ \ 00000000g0\ \ 00000000g0 \rightarrow$$

the g's denoting gliders that are definitely present. The result is a stream

$$\ldots\ 00000000gD\ \ 00000000gC\ \ 00000000gB\ \ 00000000gA \rightarrow$$

in which every information-carrying place is definitely followed by a glider g. This stream is used to kick back a full stream whose gliders are numbered:

$$\ldots\ \ldots\ldots\ldots\ldots\ldots\ldots\ \ X987654321\ \ X987654321 \rightarrow$$

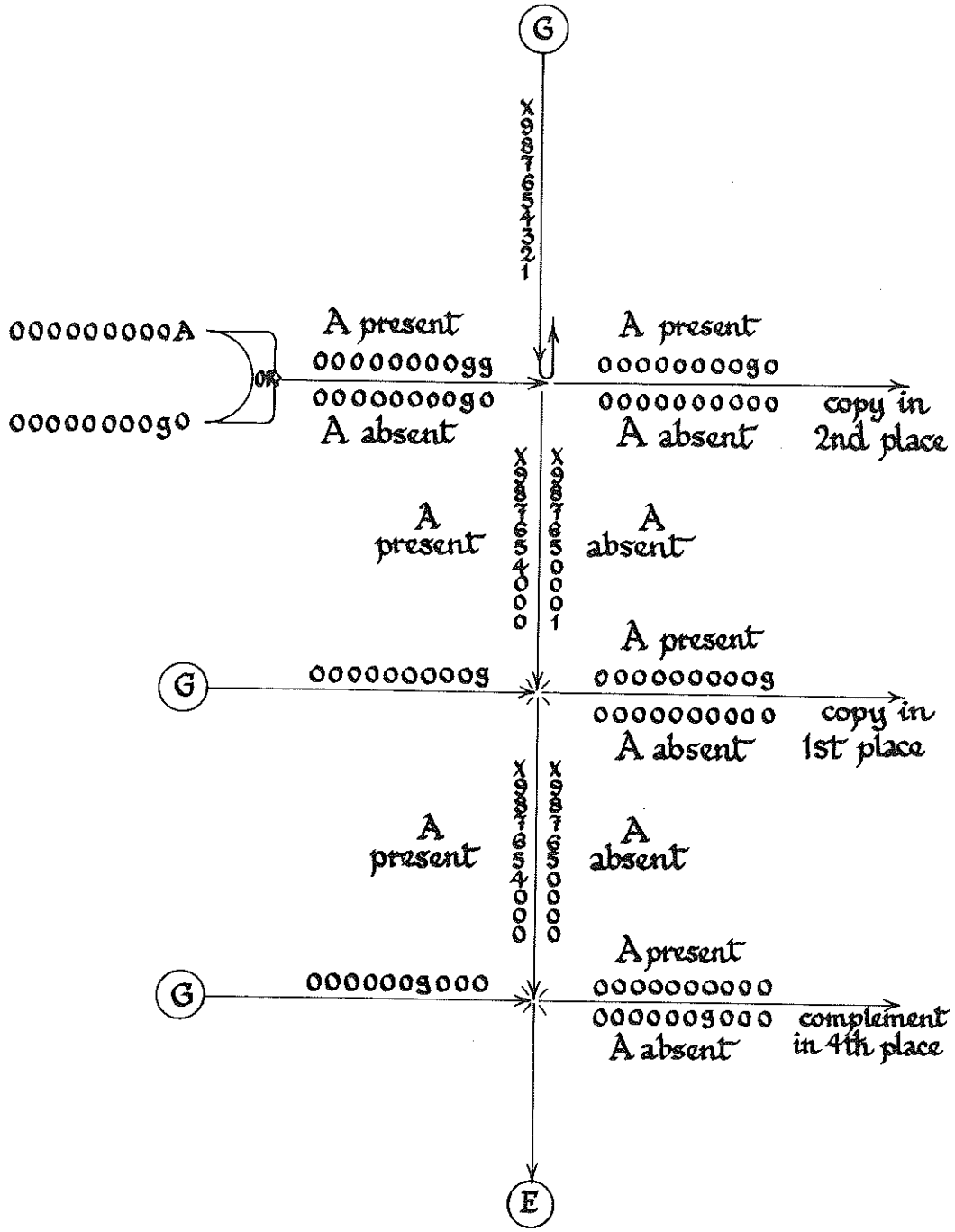


Figure 30. Copying a Glider Stream.

If glider A is *present*, it will obliterate gliders 1, 2 and 3 of the full stream and the following glider g can escape in the confusion. But if A is *absent*, then full stream glider 1 escapes and gliders 2, 3, 4 are removed instead by the following glider g. So the stream which emerges is definitely empty except for the second of every ten places and these places carry a copy of the input stream. The original full stream now manages to carry the information *twice*, in the first and fourth digits of each block, the first digit carrying the *complemented* version (which has *not* been turned through a right angle). By feeding this stream into vanishing reactions with suitably thin streams we can recover the original stream either complemented or not, and freed from undesirable accompanying gliders! Figure 30 shows these techniques in action.

From here on it's just an engineering problem to construct an arbitrarily large finite (and very slow!) computer. Our engineer has been given the tools—let him finish the job! We know that such computers can be programmed to do many things. The most important ones that we will want it to do involve emitting sequences of gliders at precisely controlled positions and times.

AUXILIARY STORAGE

Of course the engineer will probably have designed an internal memory for our computer using circulating delay lines of glider streams. Unfortunately this won't be enough for the kind of problem we have in mind, and we'll have to find some way of adjoining an *external* memory, capable of holding arbitrarily large numbers. To build this memory, we'll need an additional static piece (the block).

For instance we might ask the computer to compute

$$a^n + b^n \text{ and } c^n$$

for *all* quadruples (a,b,c,n) in turn and stop when it finds a quadruple for which

$$a^n + b^n = c^n.$$

We don't know how big a, b, c and n might get, and they'll almost certainly get too large even to be written in the internal memory.

So we're going to adjoin some auxiliary storage registers, each of which will store an arbitrarily large number. Figure 31 shows the general plan. Each register contains a block, whose distance from the computer (on a certain scale) indicates the number it contains. In the figure, register A contains 3, B contains 7, C contains 0 and D contains 2. When the contents of a register is 0, the block is just inside the computer. All we have to do is to provide a way for the computer to

increase the contents of a register by 1,
decrease the contents of a register by 1, and
test whether the contents are 0.

Fortunately each of these can be accomplished by a suitable fleet of gliders. One such fleet is off to increase register B by one! And another glider is about to discover that register C contains 0.

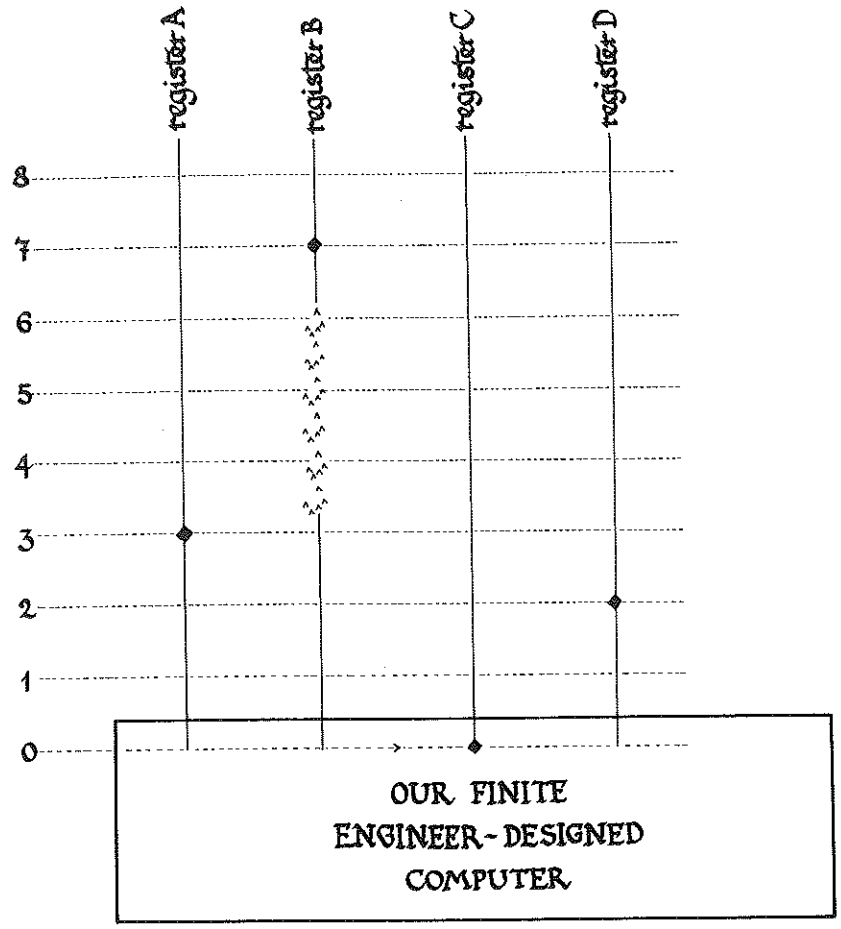


Figure 31. Auxiliary Storage.

HOW WE MOVE BLOCKS

To find these fleets we studied the six possible glider–block crashes. One of them does indeed bring the block in a bit, but unfortunately by a knight's move. However the block can be brought back onto the proper diagonal by repeating the process with a reflected glider on a parallel course. The combined effect of this pair of gliders is to pull the block back three diagonal places (Fig. 32).

Unfortunately there is no single glider-block crash which moves the block further away, but there is a crash which produces the arrangement of 4 beehives we call a honey farm, and two of these four are slightly further away, and so we can send in second, third and fourth gliders to annihilate three of the beehives, and then a fifth glider which converts the remaining beehive back into a block. The total effect again pushes the block off the proper diagonal, but a second team of five gliders will restore this, resulting in a block just one diagonal place further out! (Fig. 33).

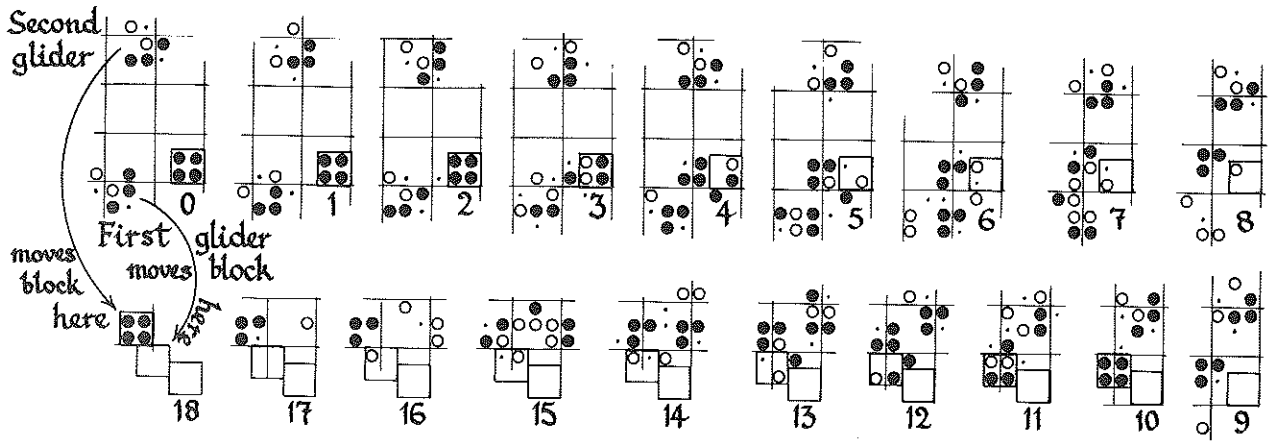


Figure 32. Two Gliders Pull a Block Back Three Diagonal Places.

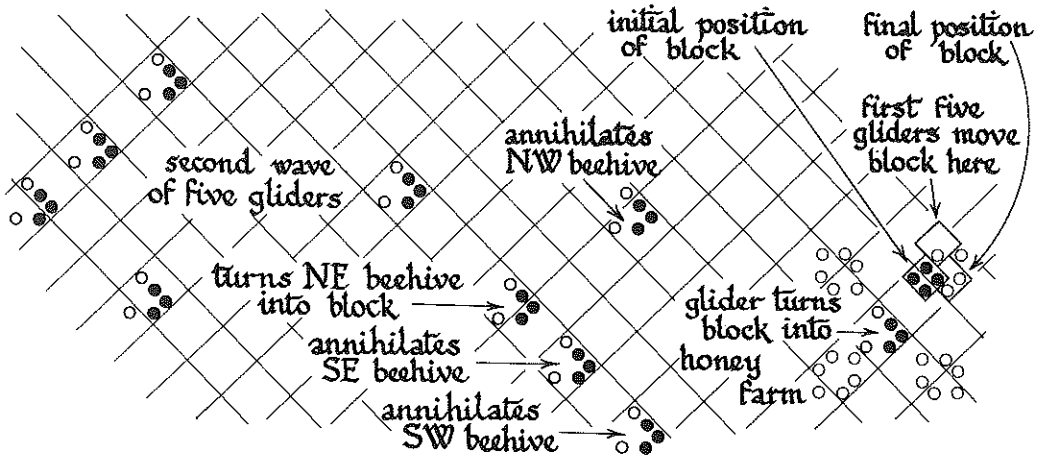


Figure 33. Ten Gliders Move a Block Just One Diagonal Place.

We therefore choose a diagonal distance of 3 to represent a change of 1 in a register and can decrease the contents of a register using a pair of gliders, or increase it using 3 flotillas of 10.

Apart from the difficulty discussed in the next section we have now finished the work, for Minsky has shown that a finite computer, equipped with memory registers like the ones in Fig. 30, can be programmed to attack arbitrarily complicated mathematical problems.

A LITTLE DIFFICULTY

But now comes the problem. Every glider in our finite computer has at some time been produced by a glider gun, so how could we arrange to send those gliders along closely parallel, but distinct paths? Surely one gun would have to fire right through another (Fig. 34)? Our technique of **side-tracking** uses three computer controlled guns G_1, G_2, G_3 as in Fig. 35. These are programmed to emit gliders exactly when we want them to.

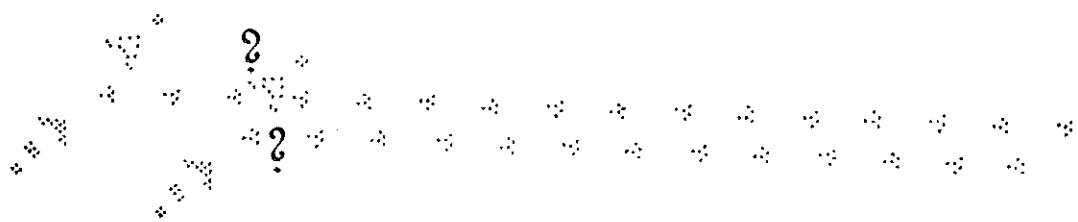


Figure 34. How Can One Gun Fire Through Another?

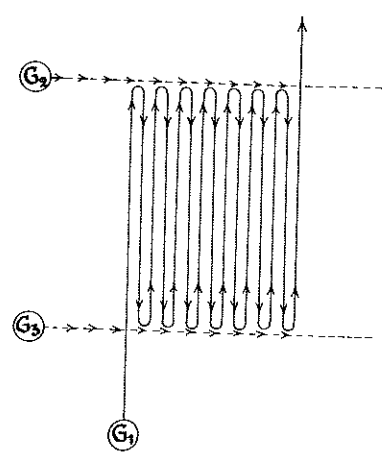


Figure 35. Side-tracking.

Firstly, G_1 emits a glider g travelling upwards,
Secondly, G_2 emits a glider at just the right time to kick g back downwards,
Thirdly, G_3 kicks g back up again,

and so on, alternately, until at a suitable time G_2 fails to fire and g is released. By controlling the number of times G_2 and G_3 fire, the *same* guns can be used to send a succession of gliders along distinct parallel paths.

MISSION COMPLETED—WILL SELF-DESTRUCT

Side-tracking can be used for a much more spectacular juggling act! We can actually program our computer to throw a glider into the air *and* bring it back down again. In Fig. 36, G_1, G_2, G_3 behave as before and can be programmed to arrange that a glider g ends up travelling Eastwards arbitrarily far above the ground. But G_4 has been arranged to emit a glider which will be kicked back down by g. We could even arrange to kick it back up again, then down again, then up again,... as suggested by the dotted lines in Fig. 36.

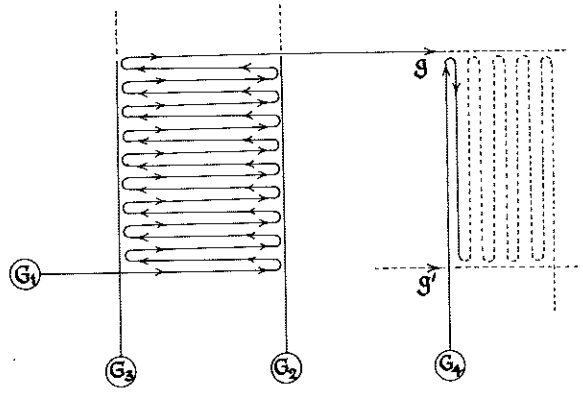


Figure 36. Double Side-tracking.

Using such techniques we can design a program for our computer which will send large numbers of gliders far out into space and then turn them round so that they head back towards the computer along precisely defined tracks (Fig. 37).

Now comes the clever part. Figures 38(a), 38(b) and 25(a) show that the eaters, the guns' moving parts, and blocks, can all be destroyed by aiming suitably positioned gliders from behind their backs. If the computer is cleverly designed we can even destroy it completely by an appropriate configuration of gliders!

Here's the idea. We design the computer so that every glider emitted by a gun or circulating in a loop would, if not deflected by meeting other gliders, be eventually consumed by an appropriately placed eater. Then we design our attacking force of gliders to shoot the computer down, guns first. After each gun is destroyed we wait until any gliders it has already emitted have percolated through the system and either been destroyed by other gliders, or swallowed by eaters, before attacking the next gun. When all the guns are destroyed we shoot down the eaters and blocks.

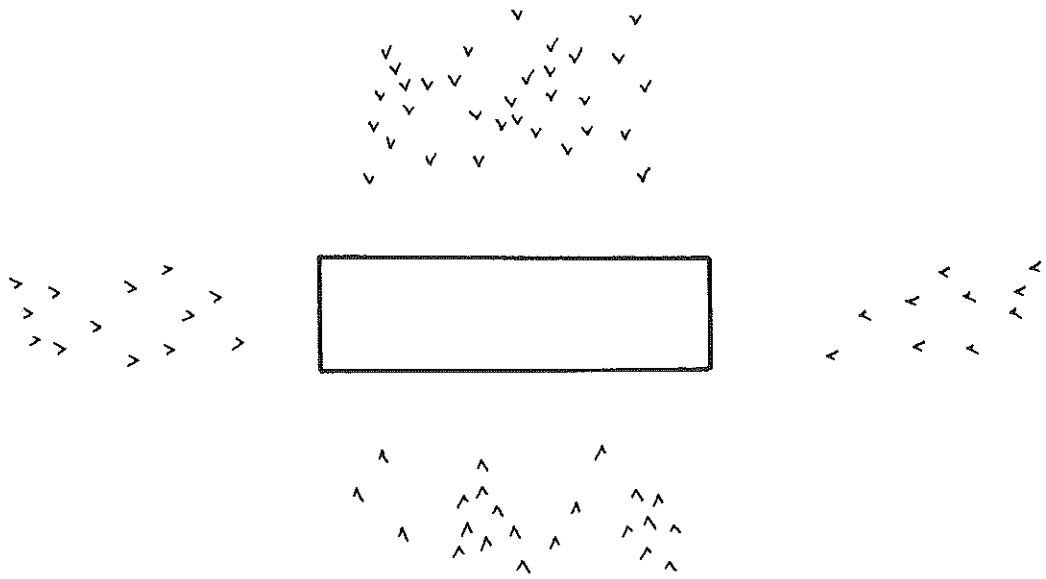


Figure 37. Self-attack from All Directions!

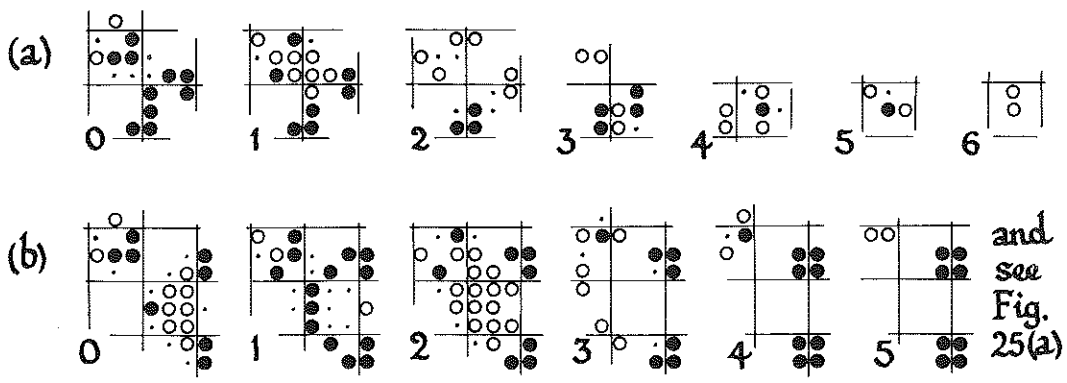


Figure 38. (a) The Eater Eaten! (b) The Gun Gunned Down!

The whole process requires some care. Each gun G_i must have a matching eater E_i, and G_i and E_i lie in a strip of the plane which contains no other static parts of the computer (Fig. 39). The gliders $g_1, g_2, g_3, \ldots$ with which we shoot down a given gun can be arbitrarily widely spaced in time provided they come in along the right tracks. Moreover we can arrange to shoot down the successive guns, eaters and blocks after increasingly long intervals of time.

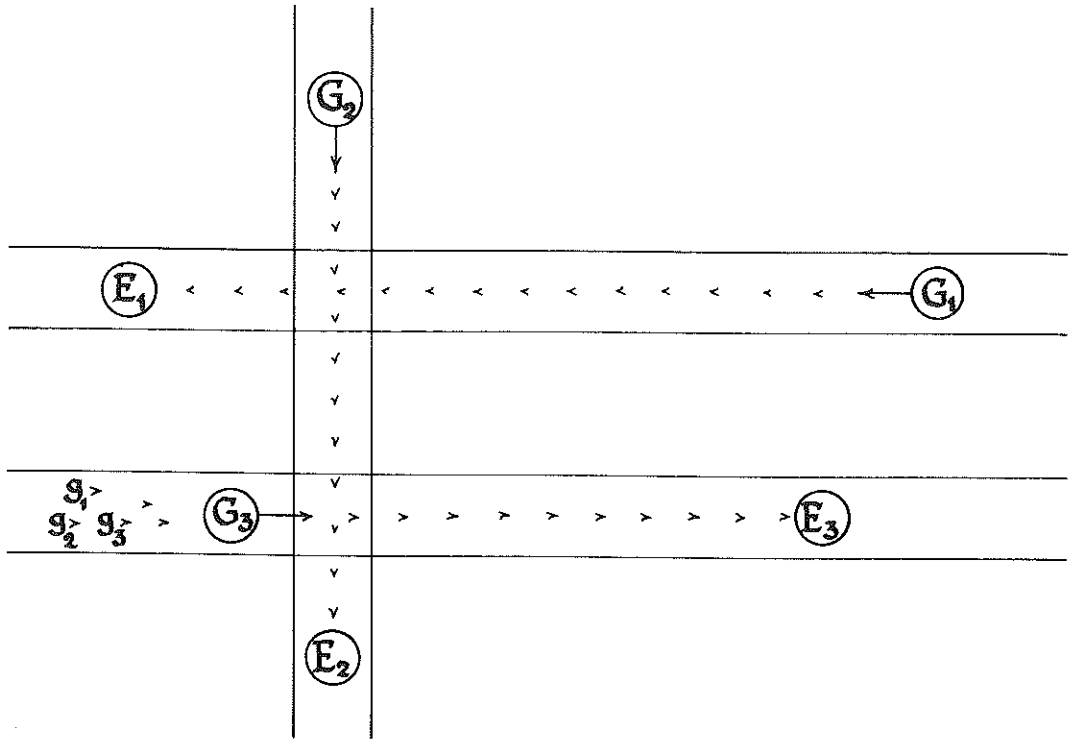


Figure 39. Arranging Destroyable Guns.

However, it *can* be done! We intend to use it like this. Program the computer to look for a solution of an arbitrarily hard problem, such as Fermat's. If it never finds a solution it will just go on forever. However, if it *does* find a solution we instruct it to throw into the air a precisely arranged army of gliders, then reduce all its storage numbers to zero (this brings all the blocks inside the computer), switch off, and await its fate. The attacking glider army, of course, is exactly what's needed to obliterate the computer, leaving no trace. It's important to realize that a *fixed* computer can be programmed to produce many different patterns of gliders and in particular the one required to kill itself. The information about this glider pattern can be held by the numbers in the memory of the computer and not in the computer's design.

Since mathematical logicians have proved that there's no technique which guarantees to tell when arbitrary arithmetical problems have solutions, there's no technique that's guaranteed to tell even when a Life configuration will fade away completely. The kind of computer we have simulated is technically known as a *universal machine* because it can be programmed to perform any desired calculation. We can summarize our result in this answer to our chapter heading:

LIFE IS UNIVERSAL!

EXTRAS

LIFE COMPUTERS CAN REPRODUCE!

Eaters and guns can be made by crashing suitable fleets of gliders, so it's possible to build a computer simply by crashing some enormously large initial pattern of gliders. Moreover, we can design a computer whose sole aim in Life is to throw just such a pattern of gliders into the air. In this way one computer can give birth to another, which can, if we like, be an exact copy of the first. Alternatively, we could arrange that the first computer eliminates itself after giving birth; then we would regard the second as a reincarnation of the first.

> There are Life patterns which behave like self-replicating animals.

> There are Life patterns which move steadily in any desired rational direction, recovering their initial form exactly after some fixed number of generations.

GENETIC ENGINEERING

We've now shown that among finite Life patterns there is a very small proportion behaving like self-replicating animals. Moreover, it is presumably possible to design such patterns which will survive inside the typical Life environment (a sort of primordial broth made of blocks, blinkers, gliders, ...). It might for instance do this by shooting out masses of gliders to detect nearby objects and then take appropriate action to eliminate them. So one of these "animals" could be more or less adjusted to its environment than another. If both were self-replicating and shared a common territory, presumably more copies of the better adapted one would survive and replicate.

WHITHER LIFE?

From here on is a familiar story. Inside any sufficiently large random broth, we expect, *just by chance*, that there will be some of these self-replicating creatures! Any particularly well adapted

ones will gradually come to populate their territory. Sometimes one of the creatures will be accidentally modified by some unusual object which it was not programmed to avoid. Most of these modifications, or **mutations**, are likely to be harmful and will adversely affect the animal's chances of survival, but very occasionally, there will be some *beneficial* mutations. In these cases the modified animals will slowly come to predominate in their territory, and so on. There seems to be no limit to this process of evolution.

> It's probable, given a large enough Life space, initially in a random state, that after a long time, intelligent self-reproducing animals will emerge and populate some parts of the space.

This is more than mere speculation, since the earlier parts are based on precisely proved theorems. Of course, "sufficiently large" means very large indeed, and we can't prove that "living" animals of any kind are likely to emerge in any Life space we can construct in practice.

It's remarkable how such a simple system of genetic rules can lead to such far-reaching results. It may be argued that the small configurations so far looked at correspond roughly to the molecular level in the real world. If a two-state cellular automaton can produce such varied and esoteric phenomena from these simple laws, how much more so in our own universe?

Analogies with real life processes are impossible to resist. If a primordial broth of amino-acids is large enough, and there is sufficient time, self-replicating moving automata may result from transition rules built into the structure of matter and the laws of nature. There is even the possibility that space-time itself is granular, composed of discrete units, and that the universe, as Edward Fredkin of M.I.T. and others have suggested, is a cellular automaton run by an enormous computer. If so, what we call motion may be only simulated motion. A moving particle in the ultimate microlevel may be essentially the same as one of our gliders, appearing to move on the macrolevel, whereas actually there is only an alteration of states of basic space-time cells in obedience to transition rules that have yet to be discovered.

REFERENCES AND FURTHER READING

Clark C. Abt, "Serious Games: The Art and Science of Games that Simulate Life", The Viking Press, 1970.

Michael A. Arbib, Simple self-reproducing universal automata, Information and Control, **9** (1966) 177–189.

E.R. Banks, Information Processing and Transmission in Cellular Automata, Ph.D. thesis, M.I.T., 71:01:15.

E.F. Codd, "Cellular Automata", Academic Press, New York and London, 1968.

Martin Gardner, Mathematical Games, Sci. Amer. **223** #4 (Oct. 1970) 120–123; **223** #5 (Nov. 1970) 118; **223** #6 (Dec. 1970) 114; **224** #1 (Jan. 1971) 108; **224** #2 (Feb. 1971) 112–117; **224** #3 (Mar. 1971) 108–109; **224** #4 (Apr. 1971) 116–117; **225** #5 (Nov. 1971) 120–121; **226** #1 (Jan. 1972) 107; **233** #6 (Dec. 1975).

M.J.E. Golay, Hexagonal parallel pattern transformations, IEEE Trans. Computers **C18** (1969) 733–740.