Physica 10D (1984) 81–95
North-Holland, Amsterdam

PHYSICS-LIKE MODELS OF COMPUTATION*

Norman MARGOLUS
MIT Laboratory for Computer Science, Cambridge Massachusetts 02139, USA

Reversible Cellular Automata are computer-models that embody discrete analogues of the classical-physics notions of space, time, locality, and microscopic reversibility. They are offered as a step towards models of computation that are closer to fundamental physics.

1. Introduction

Reversible Cellular Automata (RCA) are computer-models that embody discrete analogues of the classical-physics notions of space, time, locality, and microscopic reversibility.

In this paper, I will describe some RCA, explain how they can be used as computer models, and discuss RCA analogues of energy and entropy – concepts that are fundamental in physics, but have not played a fundamental role in computer theory.

2. Cellular automata

In CA, 'space' is a regular lattice of 'cells', each of which contains one of a small allowed set of integers. Only cells that are close together interact in one 'time-step' – the time evolution is given by a rule that looks at the contents of a few neighbouring cells, and decides what should change. At each step, this local rule is applied everywhere simultaneously[10].

The best-known example of such a 'digital-world' is Conway's[5] "Game of Life". On a sheet of graph-paper, fill each cell with a '1' or a '0'. In each three-by-three neighbourhood there is a center cell and eight adjacent cells. The new state of each cell is determined by counting the number of adjacent 1's – if exactly two adjacent cells contain a one, the center is left unchanged. If three are ones, the center becomes a one. In all other cases, the center becomes a zero.

Such a rule gives rise to a set of characteristic patterns that 'move' (reappear in a slightly displaced position after some number of steps) patterns that are stable (unchanging with time) patterns that oscillate (pass through some cycle of configurations) and many very complicated interactions and behaviours. The evolution of a given initial configuration is often very hard to anticipate (see colour plate in [9]).

One way to show that a given rule can exhibit complicated behaviour is to show (as has been done for "Life"[4]) that in the corresponding 'world' it is possible to have computers. If you start the automaton with an appropriate initial state, you will see digits acting as signals moving about and interacting with each other to perform all of the logical operations of a digital computer. Such a computer-automaton is said to be *universal*.**

* This research was supported in part by the Defense Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contracts Nos. N00014–75-C-0661 and N00014–83–K–0125, and in part by NSF Grant No. 8214312-IST.

** Von Neumann[10] was interested in the problem of evolution – could life emerge from simple rules? He exhibited a CA rule that permitted computers, and in which these computers could reproduce and mutate. In this paper, I refer only to the existence of computers when I use the term universal.

3. Reversible cellular automata

Any CA rule can be described by an equation of the form*

$$S_{i,t+1} = f(S_{\{i\},t}), \tag{1}$$

where $S_{i,t+1}$ is the state of the cell at position 'i' and at time '$t + 1$', and $f(S_{\{i\},t})$ is a function of the states of cells in a neighbourhood of i, at time t.

In general, (1) gives rise to a non-invertible dynamics. If f is the 'Life' rule, this evolution is not reversible – if an area now contains only zeros, did it contain zeros one step ago, or were there perhaps some isolated ones that just changed? Its impossible to tell.

It turns out to be very easy to write down CA laws that give an invertible dynamics – just as easy as constructing irreversible ones, in fact. Consider first the following finite difference equation, with x_t a real variable:

$$x_{t+1} = f(x_t) - x_{t-1}. \tag{2}$$

If you want to compute x_{t+1}, you must know x_t and x_{t-1} – these two constitute the complete 'state' of the system. For what functions f will the time evolution be invertible?

$$x_{t-1} = f(x_t) - x_{t+1}, \tag{3}$$

therefore any f at all will do**! Knowing x for two consecutive times allows you to calculate any preceding or any succeeding value of x (To my knowledge Fredkin[2] was the first to study reversibility in finite-difference-equations of this sort.)

The generalization to CA is straightforward – let x in (2) be replaced by c_i, the contents of the cell at position 'i' in our automaton,

$$c_{i,t+1} = f(c_{\{i\},t}) - c_{i,t-1}, \tag{4}$$

where $f(c_{\{i\},t})$ is any function involving the contents of cells near position 'i', at time 't', and the difference is taken mod the number of allowed cell values***. If we let the state of a cell correspond to its contents in two successive steps, then (4) can be reexpressed in the form (1), but its reversibility is not manifest†.

Such rules can be universal (I give an example in the appendix). Reversible computation is a relatively new idea [1, 3, 8] that has been used to show that a fundamental lower bound on dissipation in computers associated with the irreversibility of conventional logic elements[6] can be avoided.

4. Entropy in RCA

If we fill the cells of our automaton with randomly chosen binary values and then evolve it according to the Life rule, we see a complex ebb and flow of structures and activity, with so-called 'gliders' arising here and there, moving across clumps of zeros, and then being drawn back into a complex boiling 'soup' of activity, or perhaps rekindling complicated interactions in an area which had settled down into uncoupled, short period oscillating structures.

If, instead of the Life rule, we follow some invertible time evolution, we invariably find that, at each step, the state of the automaton looks just as random as when we started‡. This is expected

*This serves to clarify what sorts of systems we're dealing with, but is often not the simplest or most illuminating way to express the rule.

**Assuming integer addition and subtraction is done without error, if such an equation is iterated on a digital computer, its time evolution remains *exactly* reversible, despite roundoff and truncation errors in computing f.

***Differences mod-k and logical functions can always be re-expressed as ordinary polynomial functions. For example, if A and B are binary variables, then $(A - B)^2$ is the same as $A + B(\text{mod } 2)$, $1 - A$ is the same as not(A), $A * B$ is the same as and(A, B), etc. Thus (4) is equivalent to an ordinary real-variable finite difference equation with integer initial conditions.

†The global time evolution generated by (4) is not guaranteed to be invertible unless suitable boundary conditions are chosen, such as no boundary (i.e. an infinite or periodic space) or 'fixed' boundaries (cell values on the boundary are not allowed to change with time).

‡Spatial correlations will not arise if they are initially absent, but time correlations are often very evident, and are characteristic of the particular rule being employed – see the next section.

from a simple counting argument, since most configurations look random (only a very few random-looking initial configurations can be mapped by a given number of invertible steps into the few simple-looking configurations, since the overall mapping is bijective).

This is not meant to imply that RCA are less interesting than irreversible CA. Starting an RCA from a random state is like starting a thermodynamic system in a maximum entropy state – its not allowed to get any simpler since its randomness can't decrease, and it can't get more complicated, since its already as random as it can be, and so nothing much happens.

If we start an RCA from a very non-random state (e.g. some small pattern on a background of zeros) then we can have an interesting time evolution. If we choose a rule and an initial state that allow information to propagate, then what tends to happen is that the state of the RCA becomes more and more complicated. More precisely, if each state of the automaton is viewed as a 'message', with the contents of the cells being the characters of the message, and if only local measures of correlation are applied, then the amount of information* in successive messages is increasing. Of course the automaton is really only repeatedly encrypting its state, and so if all correlations are taken into account the amount of information really never changes. What happens is that the automaton will introduce some redundancy into the message, and use more cells to encode the same information. Information that was initially localized becomes spread out as correlations between the states of many cells, and it becomes very difficult for a locally invertible evolution to put the redundant pieces back together**. To use an analogy, an invertible mapping could change two copies of this document into one copy, and several sheets of blank paper. Two separate invertible mappings, each acting only on one of the copies, could not accomplish this end.

From the point of view of creatures 'living' inside an RCA, their inability to make use of complicated correlations between large numbers of cells means that for all practical purposes, the entropy of the automaton increases. To use a thermodynamic analogy, if I want to compress a gas, it doesn't help me to know that the gas was all in one corner of the room just a few minutes ago. I have no ability to make use of the complicated correlations that this statement implies, and so I say that entropy increased when the gas expanded to its current volume.

5. Conservation laws in second-order RCA

In general, an RCA has as many conserved quantities as there are cells – it 'remembers' the initial state of each cell, since you can recover this information by running the system backwards. Do these give rise to any invariants which can be computed in a local manner from the current state of the system? Can we find an invariant that is analogous to a classical mechanical energy?

In RCA, the simplest locally-computable invariants are of course cells whose values never change***. Such situations can arise because many rules ignore the remainder of the neighbours when part of the neighbourhood has some particular configuration†. For example, consider any rule that, in all cases where the center cell of the neighbourhood is 1, ignores the rest of the neighbours and returns a 2. Such a rule, when used with (4), results in a very simple conservation law. If we look at the case in one dimension where the automaton at two consecutive time-steps looks like

*For a discussion of the information content of a message, cf. [7].

**(4) generates a locally invertible time evolution. If we know the values of cells near position i at two successive times, we can tell what the preceding value of the center cell was.

***In mechanics, this corresponds to degrees of freedom that, for certain initial conditions, are decoupled from the rest.

†For rules with 2 states per cell, only two rules, "count the parity of the neighbourhood" and its complement, have no configuration of part of the neighbourhood that makes the remaining neighbours irrelevant.

this:

```
t − 1   ...1...   '.' indicates a cell whose value is
t       ...1...   irrelevant to the discussion   (5)
```

then the center cell here will always be a 1.

For a more interesting 1-dimensional example, consider a 2 state per cell CA with a rule f that returns a 1 iff each of the two cells adjacent to the center is the same as the center:

$$f(c_{\{x\},t}) = \begin{cases} 1, & \text{if } c_{x-1,t} = c_{x,t} = c_{x+1,t} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

With this rule, 'a' and 'b' standing for any binary values, and '$\bar{a}$', '$\bar{b}$' their binary complements, the second-order time evolution given by (4) says that

$$\left\{\begin{array}{ll} t-1 & \ldots a\bar{a}\ldots \\ t & \ldots b\bar{b}\ldots \end{array}\right\} \rightarrow \left\{\begin{array}{ll} t & \ldots b\bar{b}\ldots \\ t+1 & \ldots a\bar{a}\ldots \end{array}\right\} \tag{7}$$

which is again of the same form, so these two cells are decoupled from the rest of the automaton. Any cell which is *not* initially part of such a pair will never be (and never was)*; counting all such cells gives us an (invariant) estimate of how many cells are available to represent dynamically changing information (but only an estimate – whole regions may be decoupled from the rest of the automaton because they are surrounded by a wall of decoupled cells**; a local counting wouldn't reveal this).

If we concentrate on the active (as opposed to the decoupled) cells, we can distinguish various kinds of activity, and try to associate conserved quantities with each. As a simple 1-dimensional example, consider 'dislocations' propagating in a regular background pattern of cells. Using the rule (6) again, consider the sequence of steps shown in fig. 1 (light and dark squares stand for 0's and 1's respectively; dislocations are triplets in a background of pairs and are outlined for emphasis). In this evolution, the number of such 'signals' is the

* In irreversible CA, a guarantee that a cell will always be part of such a pair does not guarantee that it always has been.

** An extreme instance of 'decoupling' of entire regions occurs with any rule that doesn't depend on the center cell, but depends on its nearest neighbours. For example, in 1D we might have a region that looks like this:

```
t − 1   ... 1.1.0.1...
t       ...1 .0.0.1.1...
t + 1   ... ?.?.?.?...
```

From (4) it is clear that we have enough information to compute the states of the cells marked with '?' – the system decouples into two entirely independent (but interleaved) sublattices, each evolving without reference to the other.

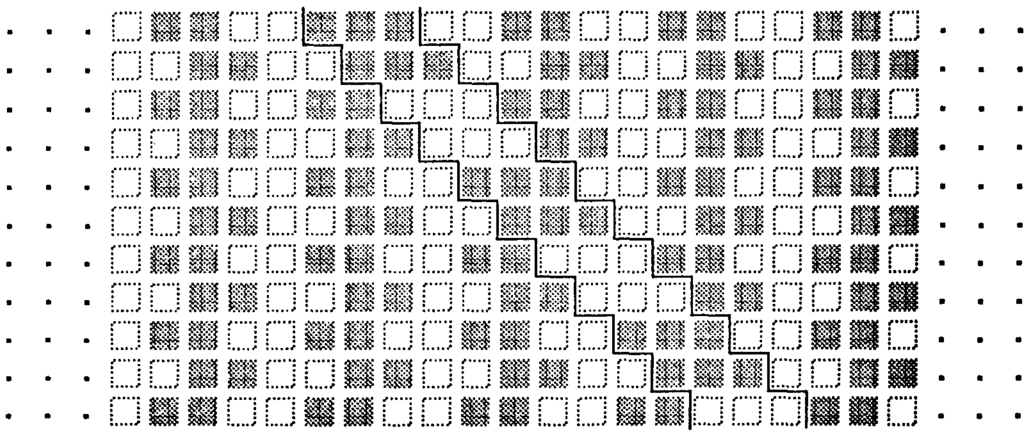


Fig. 1.

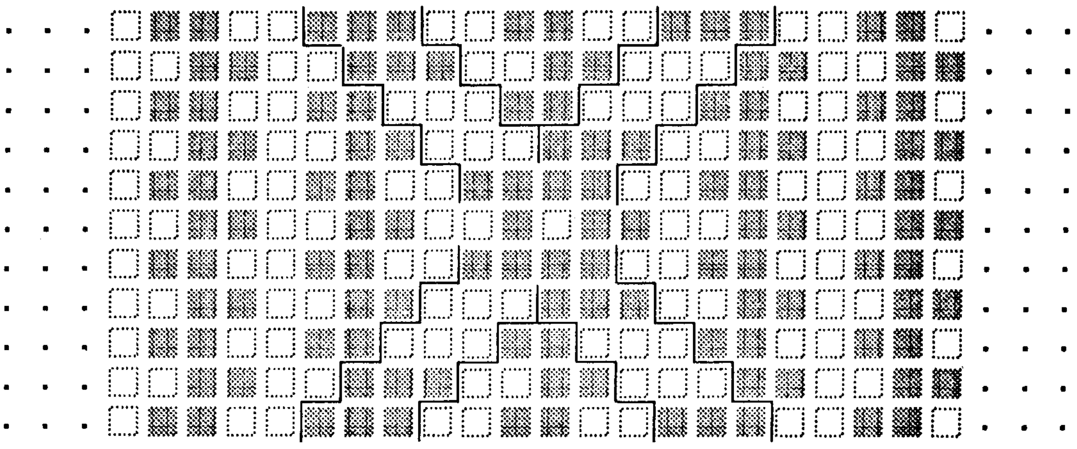


Fig. 2.

same as the number of blocks of cells we start off with the form $\begin{smallmatrix}\cdots & aa & \cdots\\ \cdots & bb & \cdots\end{smallmatrix}$, and is conserved.

If two such dislocations collide, we know that they can't just completely stop moving. Proof: if we tried to invert the evolution, we wouldn't know when to start the signals moving again. In fig. 2, the following quantity is the same for every pair of consecutive time-steps:

$$\left(\#\text{ of cells in blocks of form }\begin{matrix}\ldots aa\ldots\\ \ldots bb\ldots\end{matrix}\right)$$
$$+\left(\#\text{ of cells in blocks of form }\begin{matrix}\ldots c.c\ldots\\ \ldots d.d\ldots\end{matrix}\right) \qquad (8)$$

The first term counts the number of moving signals, and so could be thought of as a 'kinetic-energy' analogue. The second term accounts for the disappearance of this 'K.E.' during a collision, and so could be considered a potential-energy analogue.

6. First-order RCA

Rather than continue to analyze RCA in order to discover conservation laws, we will now proceed to construct a class of automata that all obey a very simple local conservation law: the total number of 1's never changes, and neither does the number of 0's.

The trick we will use is quite general, but it will be illustrated in 2 dimensions with 2 states per cell. Fig. 3 shows a Cartesian lattice of cells, divided into 2 × 2 blocks of cells. We treat each 2 × 2 block as a conservative-logic[3] gate, with 4 inputs (its current state) and 4 outputs (its next state). These 'gates' are interconnected in an entirely uniform and predictable manner – in applying the rule to the 2 × 2 blocks, we alternate between using the solid blocking in this diagram for one step, and then using the dotted blocking for the next*. Fig.

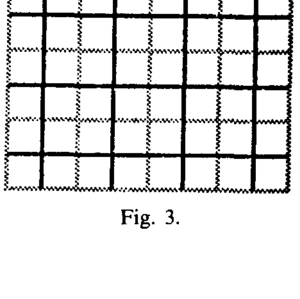


Fig. 3.

* If f_s is the global rule that applies to the solid blocking, and f_d to the dotted blocking, then $S_{t+1}=f_s(f_d(S_t))$ describes the evolution using a time independent rule. By including a small amount of positional information in the state of each cell, this rule can be written in the form(1).

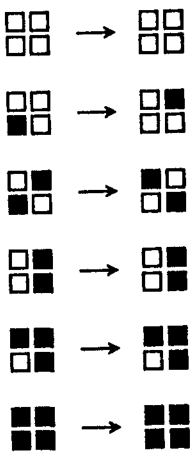


Fig. 4.

4 shows an example of a conservative rule (one that conserves 1's and 0's) that is reversible. In the case of all 0's or all 1's, there is no choice, they remain unchanged. Any rotation of one of the blocks on the left is mapped onto the corresponding rotation of the result to its right – this rule is rotationally symmetrical, and these are all of the possible cases. Since each distinct initial state of a block is mapped onto a distinct final state, this rule is reversible. As will be shown later, the automaton corresponding to this rule is universal.

One could easily have written down an example of a rule that conserved 1's and 0's, but that didn't always map a distinct initial state of a block into a distinct final state – such a rule would be conservative, but not reversible. As the corresponding automaton evolved, it would forget all sorts of details about the initial state, but it would always remember the numbers of 1's and 0's.* Thus the existence of an interesting local conservation law does not depend on the rule being reversible!

In the language of digital logic, a gate from which it is possible to construct any boolean function of any number of input variables is a *universal* gate. If a logic gate is not universal, then no interconnection of such gates can be a computer. Thus the only candidates for universal CA's in the scheme described above are those whose rule corresponds to a universal logic gate.

In order to promote the CA rule of fig. 4 as a link between classical mechanics and computation, I will first discuss Fredkin's Billiard Ball Model (BBM) of computation[3] – a classical mechanical system that can be used to do digital computation. It will then be easier to discuss this rule, which I call the BBMCA – a purely digital model of computation which is closely related to the BBM.

* For each gate (block), we can tell after each step how many possible predecessors the result-block has. Thus we can count exactly how much information is lost at each step.

**The BBMCA models space as being uniformly filled with gates, and so a connection is already apparent.

7. The billiard ball model of computation

The BBM is a classical mechanical system, and obeys a continuous dynamics – positions and velocities, masses and times are all real variables. In order to make it perform a digital computation, we make use of the fact that integers are also real numbers. By suitably restricting the initial conditions we allow the system to have, and by only looking at the system at regularly spaced time intervals, we can make a continuous dynamics perform a digital process. In this case, we begin with a 2-dimensional gas of identical hard spheres. If the center of a sphere is present at a given point in space at a given point in time, we will say that there is a '1' there, otherwise there is a '0' there. The 1's can move from place to place, but their number never changes.

The key insight behind the BBM is this: *every place where a collision of finite-diameter hard spheres might occur can be viewed as a boolean logic gate***. What path a ball follows depends upon whether or not it hits anything – it makes a decision.

To see how to use this decision to do boolean logic, consider fig. 5. At points A and B and at time t_i, we either put balls at A, B, or both, or we put none. Any balls present are moving as indicated with a speed 's'. If balls are present at *both* A and B, then they will collide and follow the outer

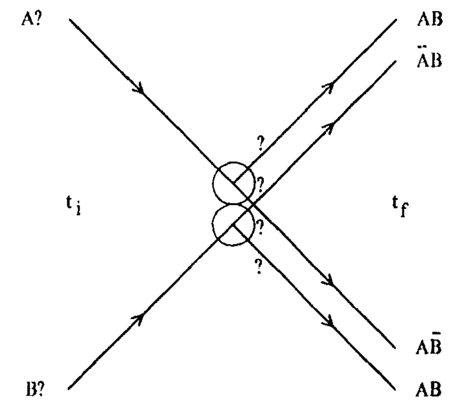


Fig. 5.

outgoing paths. Otherwise, only the inner outgoing paths will be used. At time $t = t_i$, position A is a 1 if a ball is there, and 0 otherwise (similarly for position B). At $t = t_f$, the four labeled spots have a ball or no ball – which they have is given by the logical function labeling the spot. For example, if $A = 1$ and $B = 0$, then the ball coming from A encounters no ball coming from B, and ends up at the point labeled "A and not B". A place where a collision might occur acts as a reversible, universal [3] 1-conserving logic-gate, with two inputs and four outputs. A path that may or may not contain balls acts as a signal-carrying wire. Mirrors (reflectors) allows bends in the paths. In order to be able to use the outputs from such a collision-gate as inputs to other such gates, we need to very precisely control the angle and timing of the collisions, as well as the relative speeds of the balls. We make this simple to do by severely restricting the allowed initial conditions. Each ball must start at a grid point of a Cartesian lattice, moving 'along' the grid in one of 4 allowed directions. See fig. 6. All balls move at the same speed. The time it takes a ball to move from one grid point to another we call our unit of time. The grid spacing is chosen so that balls collide while at grid-points. See fig. 7. All collisions are right angle collisions, so that one time-step after a collision, balls are still on the grid. Fixed mirrors are positioned so that balls hit them while at a grid point, and so stay on the grid. See fig. 8. By using mirrors, signals can be

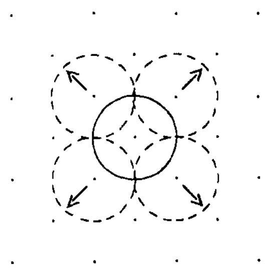


Fig. 6.

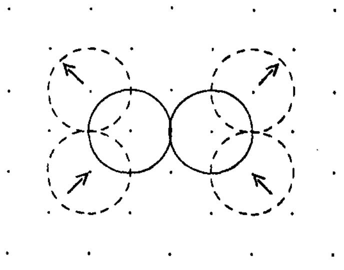


Fig. 7

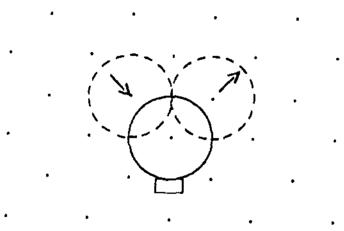


Fig. 8.

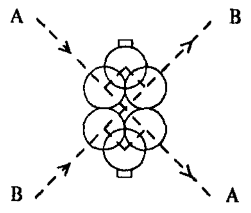


Fig. 9.

routed and delayed as required to perform digital logic. The configuration of mirrors in fig. 9 solves the problem of making two signals cross without affecting each other. (Notice that if two balls come in together, the signals cross but the balls don't!).

Mirrors and collisions determine the possible paths that signals may follow ('wires'). In order to ensure that all collisions will be right-angle collisions (and not head-on, for example, which would take us off our grid) we can label all 'wires' with arrows, and restrict initial conditions and interconnections so that a ball found on a given 'wire' always moves in the labeled direction.

Thus our universal gates can be connected as required to 'build' a computer. Computations can be pipelined – an efficient 'assembly-line' way of doing things, where questions flow in one end and finished products (answers) flow out the other, while all the stages in between are kept busy. Reversibility turns out not to be a great hindrance – unwanted intermediate results can be mostly 'erased' by copying the answer once you have it, and then running the computation backwards to get rid of everything but a copy of the inputs.

This then, in brief, is the BBM. Kinetic energy is conserved, since all collisions are elastic. Momentum is not conserved, since the mirrors are assumed to be fixed (infinitely massive).

8. The BBM cellular automaton

When viewed only at integer time-steps, the BBM consists of a Cartesian lattice of points, each of which may 'contain' a 0 or a 1, evolving according to a local rule. It would therefore seem to be a straightforward matter to find a CA rule that duplicates this digital time evolution.

Unfortunately, the most direct translation of the BBM into a CA has several problems. First of all, to have separate states of a cell to represent 4 kinds of balls (4 directions) an empty cell and a mirror, and to have the balls absolutely conserved (as they are in the original BBM) would require a standard "change the center cell" rule with 6 states per cell, and a 17 cell neighbourhood. Such a rule has a very large number of possible configurations for its neighbourhood, which makes it unwieldy. Moreover, many of these configurations involve such events as head-on collisions, which were disallowed in the BBM – a CA rule, however, should be defined for all configurations. It is not at all clear how to extend the BBM rule to these extra cases, and still have it remain reversible and 'energy' conserving.

At the expense of making collisions cause a slight delay, we can get away with the very simple rule of fig. 4, which involves only 2 states per cell in a 4 cell neighbourhood, is reversible, and conserves the number of ones (and zeros) in all cases. The $\begin{smallmatrix}□&□\\■&□\end{smallmatrix} \longrightarrow \begin{smallmatrix}□&■\\□&□\end{smallmatrix}$ (and rotations) case in fig. 4 is the one that causes an isolated '1' to propagate in a straight line, in one of four directions (depending on which of the four corners of its starting block you put it in). See fig. 10. The legend "solid" or "dotted" below each of these automaton configurations tells you whether the grouping of cells into blocks for the next application of the rule

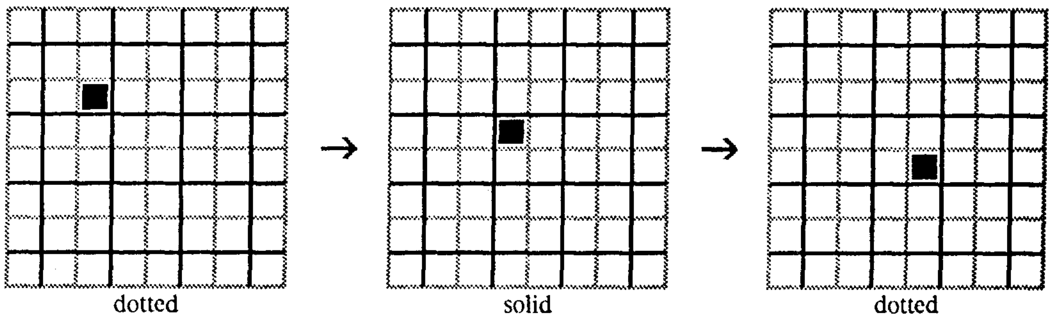


Fig. 10.

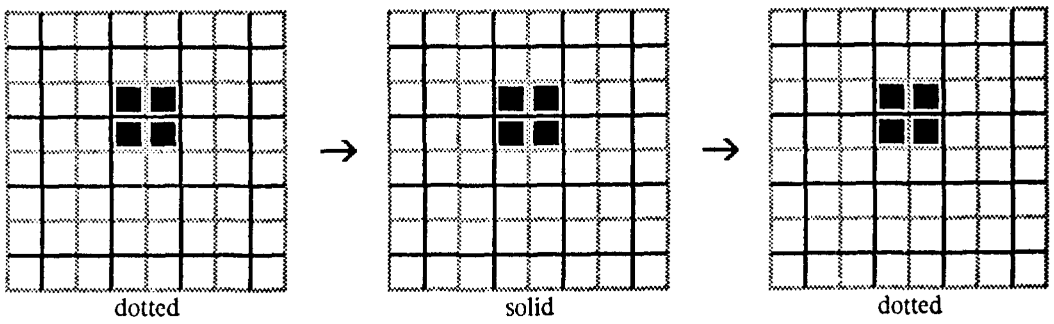


Fig. 11.

is indicated by the solid or the dotted lines. In the diagrams, a 0 is shown as an empty (blank) cell, a one is shown as a filled-in cell. Since $\begin{smallmatrix}\square&\blacksquare\\\square&\blacksquare\end{smallmatrix} \longrightarrow \begin{smallmatrix}\square&\blacksquare\\\square&\blacksquare\end{smallmatrix}$ (and rotations), a square of four ones straddling the boundary of two adjacent blocks will be stable–we will use such squares to construct mirrors. See fig 11. The four 1's straddle two dotted blocks horizontally, then two solid blocks vertically, and then two dotted again. Since $\begin{smallmatrix}\square&\blacksquare\\\blacksquare&\square\end{smallmatrix} \longrightarrow \begin{smallmatrix}\blacksquare&\square\\\square&\blacksquare\end{smallmatrix}$ (and rotations), pairs of travelling ones perform a billiard-ball type collision. See fig. 12. In all of these figures, the paths the ones were originally following have been lightly drawn in, to show that the 'and' case shown results in an outward displacement, just as in the BBM. (Unlike the BBM, there is a delay in such a collision, which we'll have to worry about in synchronizing signals). Finally, $\begin{smallmatrix}\blacksquare&\blacksquare\\\square&\blacksquare\end{smallmatrix} \longrightarrow \begin{smallmatrix}\blacksquare&\blacksquare\\\square&\blacksquare\end{smallmatrix}$ (and rotations) permits the reflection of double signals by a mirror. See fig. 13. The 'mirror' consists of two adjacent stable squares (notice that a square is stable no matter what you put next to it – its

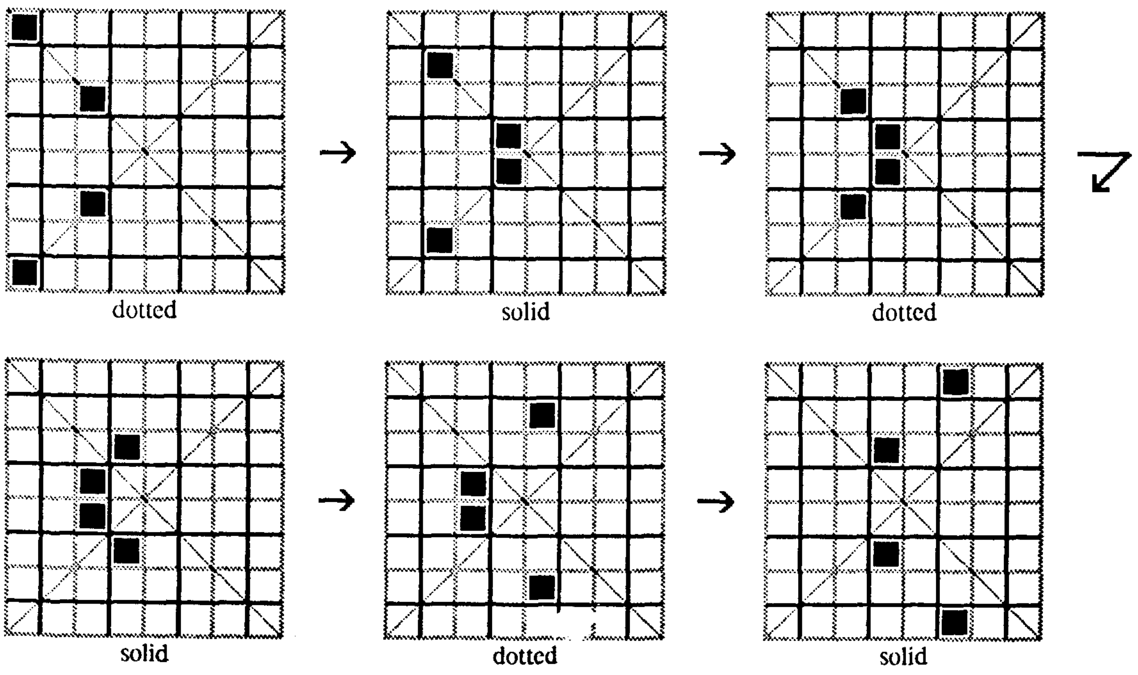


Fig. 12.

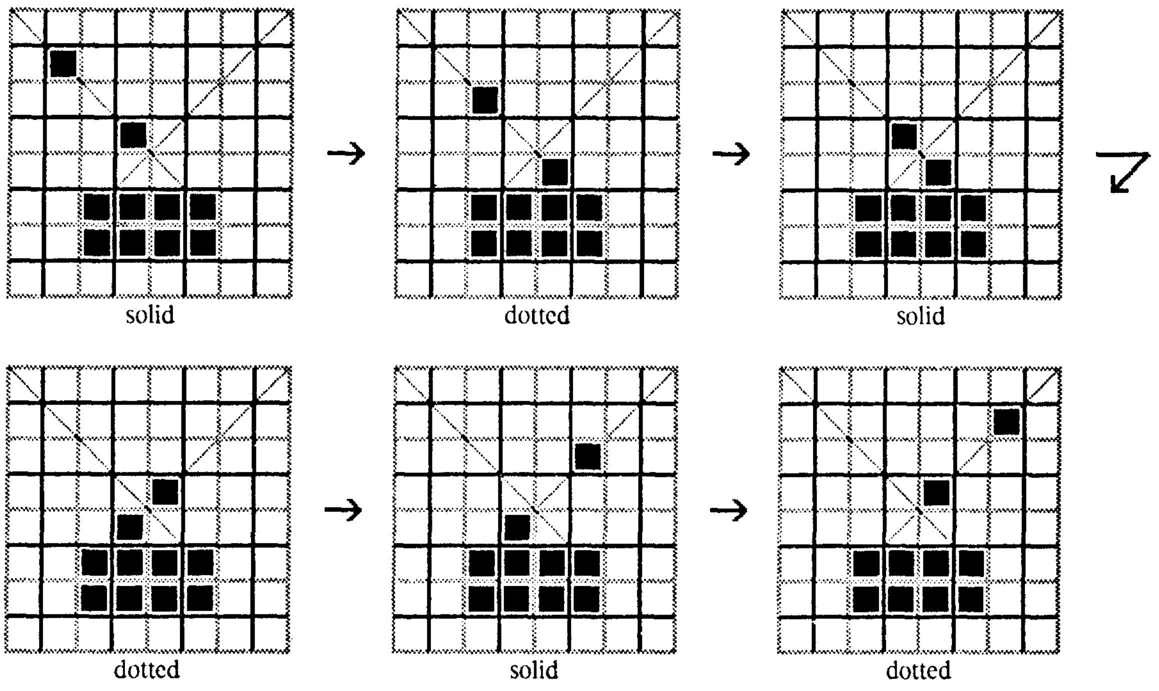


Fig. 13.

'decoupled'). Again, the signal path has been lightly drawn in. After each reflection such as that shown above, the signal has been delayed by a distance of one block along the plane of the mirror (in this picture, the signal winds up one block-column behind where it would have been had it not hit the mirror).

* We can tell how many steps a signal will take to traverse a given path (from one position where the signal is moving freely to another) by simply drawing the path joining the two points (including all points that may be visited by at least one '1') and counting how many cells are on the path.

In the BBM, such a reflection would cause no horizontal delay. We can compensate for such extra delays, as well as add any desired horizontal delay of 2 or more blocks. See fig. 14.

Suppose we want to arrange for two signals to collide, with the plane of the collision being horizontal. If we get the two signals aligned vertically and they are approaching each other as they move forward, they will collide properly. We can adjust the time it takes one or both signals to reach a given vertical column by using delays such as those in fig. 14*.

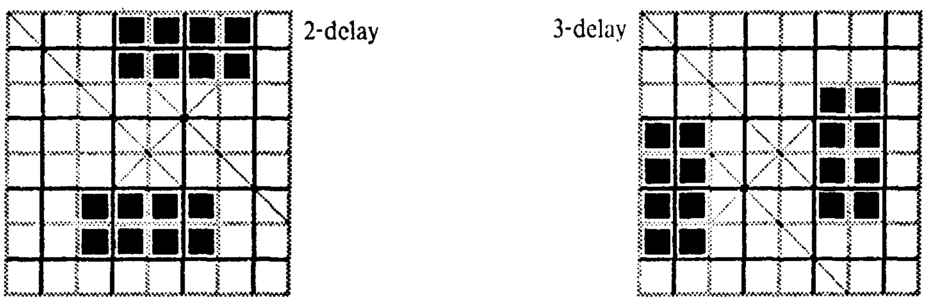


Fig. 14.

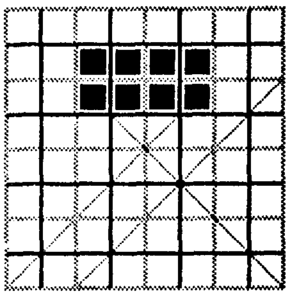


Fig. 15.

In order to allow signal-paths to cross without interacting, we use signal timing. By leaving a gap long enough for one signal (2 blocks) between all signals, we need only delay one of the paths by 2 blocks along the plane of the collision we're avoiding, in order to allow the signals to pass each other harmlessly. This gap is also enough to allow us to separate parallel output paths from a collision. See fig. 15. After the collision (fig. 12) the upper path already has a 1-block horizontal delay relative to the lower path. The mirror introduces a further 1-block delay, and so the upper signal passes through the timing-gap left in the lower signal path. With the addition of some extra synchronization and crossover delays, any BBM circuit can now be translated into a BBMCA circuit. Since the BBM has been shown to be a universal computer, the BBMCA is also.

There are many rules similar to the BBMCA that are also universal – for example, if we take the BBMCA rule of fig. 4 and modify it so that for each case shown, the result (right-hand side) is rotated 90-degrees clockwise on the 'dotted' steps, and counterclockwise on the 'solid' steps (i.e. $\begin{smallmatrix}■&□\\□&□\end{smallmatrix} \rightarrow \begin{smallmatrix}□&□\\■&□\end{smallmatrix}$ on dotted steps, and $\begin{smallmatrix}■&□\\□&□\end{smallmatrix} \rightarrow \begin{smallmatrix}□&■\\□&□\end{smallmatrix}$ on the solid steps, etc.) then we get another rule that is also computation universal. Its universality can be shown in a direct manner by using this rule to simulate the BBMCA (this rule can simulate a given BBMCA computation isomorphically using eight times as much space, and four times as much time)*.

*The idea for this BBMCA variation arose out of a discussion with Tommaso Toffoli.

9. Relationship of BBMCA to conservative logic

The collision-gate of fig. 5 has two inputs and four outputs. If we wish to consider it to be a conservative-logic gate (one that conserves both 0's and 1's) then we must regard it as a gate with four inputs and four outputs, two of the inputs being constrained to always be zeros.

The gate upon which the BBMCA is based also has four inputs and four outputs. Is there some connection here? Let us redraw the BBMCA rule in a different form (see fig. 16). The mapping of input variables onto output variables of the BBMCA has been redrawn as if the inputs all arrive and leave in a vertical column. If we use this correspondence to draw the four possible cases with $a = d = 0$, drawing □ for 0, ■ for 1, and showing each input/output case, we get an evolution (see fig. 17) which is logically the same as the collision gate. Thus the BBMCA rule of fig. 4 can be regarded as a completion of the collision-gate to a (reversible) conservative-logic gate!

```
                          a -|   |- A
ab    AB                  b -|   |- B
cd →  CD     becomes      c -|   |- C
                          d -|   |- D
```

Fig. 16.

```
□-|  |-□        □-|  |-□        □-|  |-□        □-|  |-■
□-|  |-□        ■-|  |-□        □-|  |-■        ■-|  |-□
□-|  |-□        □-|  |-■        ■-|  |-□        ■-|  |-□
□-|  |-□        □-|  |-□        □-|  |-□        □-|  |-■
```

Fig. 17.

10. Energy in the BBMCA

In the BBM, the kinetic energy is proportional to the number of moving 1's. In the BBMCA, if we let $p_{x,y,t\text{-}1/2} = c_{x,y,t} - c_{x,y,t-1}$, then $\Sigma_{xy}(p^2_{x,y,t\text{-}1/2}/2)$ counts the number of moving ones (each moving one disappears from one cell, and appears in another, so $\Sigma_{xy}p^2$ – which counts how many places change – would count each moving one twice).

The ones that aren't moving are at those places that were a one at $t-1$, and still are one at time t. Thus the number of stationary ones is $\Sigma_{xy}c_{x,y,t}c_{x,y,t-1}$. A complicated way of writing the (constant) total number of ones is

$$E_{t\text{-}1/2} = \sum_{xy} \frac{p^2_{x,y,t\text{-}1/2}}{2} + \sum_{xy} c_t c_{t-1}. \qquad (9)$$

During a collision, some of the 'kinetic-energy' changes into 'potential-energy', and then it changes back again.*

Since (9) is a constant for *any* rule for which $\Sigma_{xy}c^2_{x,y,t}$ is constant, it is not possible to derive the particular rule from this expression. We might (for example) introduce the rule into (9) by using it to eliminate c_t (thus writing E as a function of $p_{t\text{-}1/2}$ and c_{t-1}) and see if we can push the mechanics analogy further.

Using number-of-ones to play the role of energy in BBMCA circuits and considering circuits for which we have only a statistical knowledge of what the different inputs will be, elaborate thermodynamic analogues can be established, but this will be discussed elsewhere. Although the overall system has a single deterministically evolving state, from the point of view of small pieces of the system, their inputs may appear random.

11. Conclusion

The laws of nature are the ultimate computing resource — the most efficient computation imaginable would make the most direct possible use of the physical interactions and degrees of freedom available. Physical quantities and concepts would have a direct computational interpretation. Computer scientists cannot hope to find the right quantities to use to talk about efficient computation until they have models of computation that are much closer to fundamental physics. Reversible Cellular Automata are offered as a step towards this end.

* One can think of mechanical models of the BBMCA for which the two terms of (9) are proportional to the physical kinetic and potential energy of the system midway between two steps.

Appendix A

A second-order, reversible, universal automaton

This appendix describes another BBM-type automaton. As before, we begin with a 2-dimensional cartesian lattice, this time with 3 states per cell, which we can designate as -1, 0, $+1$, and which we will draw as '\', blank, and '/' respectively in diagrams.

The time evolution will be given by $c_{x,y,t+1} = f(c_{\{x,y\},t}) - c_{x,y,t-1}$, where $f(c_{\{x,y\},t})$ is a function that 'looks' at the 3×3 neighbourhood with $c_{x,y}$ as its center cell, and '$-$' is taken mod3.

For each possible configuration of the neighbourhood, f will return a value of -1, 0, or $+1$. Just as head-on collisions never arise in BBM computations, many configurations of this RCA need not arise in order to 'build' a universal computer. We will leave these cases undefined – each choice for these undefined cases defines a distinct universal RCA.

An isolated '/' or '\' will correspond to a travelling billiard ball – if only the cases defined here arise, the number of such 'balls' will be conserved. An isolated '/' will propagate along a positively sloped diagonal – its evolution will be governed by the following cases:

```
000  000  /00  000  0/0  000  000  000
000  0/0  000  000  000  /00  00/  000
000  000  000  00/  000  000  000  0/0
```

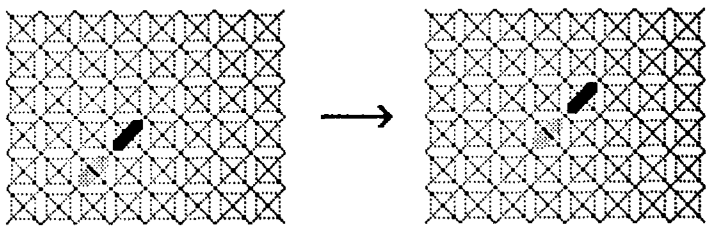


Fig. 18.

all return a '0' as the value for *f*;

```
00/  000
000  000
000  /00
```

both yield a value of '/' (i.e. + 1). A sample time evolution (using halftones to show a cell's contents at time $t-1$ and solid lines for time t, with diagonals lightly drawn through all cells) is shown in fig. 18. Intuitively, this rule at time *t* tries to make the '/' travel both forwards and backwards along its diagonal–subtracting away a '/' where it was at time $t-1$ just leaves a '/' in the forwards direction.

We define this rule to be rotationally symmetric. It will be helpful to adopt the following convention: the 90-degree clockwise rotation of

```
000           \00
000  → /  is  000  → \.
/00           000
```

Inversions are defined analogously. Thus an isolated '\' will follow a negatively sloped diagonal path if the propagation of signals is governed by the cases:

```
000  000  /00  0/0
000  0/0  000  000  → 0,
000  000  000  000
```

```
000
000  → /
/00
```

(and rotations and inversions).

For compactness in writing the complete rule, we adopt the convention that inversions as well as rotations of the cases given are mapped onto the corresponding inversions or rotations of the result given.

These cases become zero:

```
\\\ \\0 \\0 \\0 \\0 \\/ \\/ \0\ \0\ \0\ \00 \00 \00
000 000 /00 /00 //0 000 /0/ 000 /00 /00 000 /\0 /\/ ,
/// //0 000 00/ 00/ //\ /\\ /0/ 000 00/  /00 000 000

\00 \00 \00 \00 \00 \00 \00 \00 \00 \00 \00 \00 \00
/0\ /0\ /00 /00 /00 /00 /00 /00 /0/ /0/ //\ //0 //0 ,
000 00/ \00 \0/ 000 00/ 0/0 0// 000 00/ 000 000 00/

\0/ \0/ \0/ 00\\ 0\\ 0\0 0\0 0\0 0\0 00\ 00\ 00\ 00\
000 /0\ /0\ 000 000 000 000 000 /0/ 0\0 00\ 000 000 ,
/0\ \0/ 000 000 0// 000 00/ 0/0 0\0 000 00/ 000 00/

00\ 000 000 000 /\\ /\\ /0\
00/ 0\0 000 /0\ 000 /0/ 000
000 000 000 \0/ \// \\/ \0/
```

These cases become one:

```
\00 \00 \00 \00 \0/ \0/ \0/ \/\ 0\0  0\/ 0\/ 0\/
/\0 /00 /00 /00 /0\ /00 /00 /00 /\0  \00 00\ 000 ,
\/0 /\0 /00 /0/ 00/ 000 00/ 000 \/0  /00 000 /00

00\ 000 000 000 000 000 000
/\0 000 000 /\\ /\0 /\0 /\/
\/0 /00 /0/ \// \/0 \// \/0
```

(plus rotations and inversions). There are 2617 undefined cases.

Using this rule, a mirror is shown in fig. 19. We needed to define certain cases just to allow a mirror to remain unchanged when no signals are nearby.

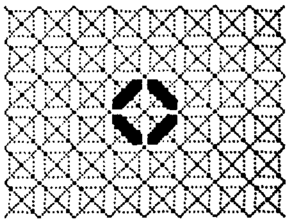


Fig. 19.

A signal bouncing on a mirror is shown in fig. 20. (Notice that there is no horizontal delay, as there was in the BBMCA). If this signal had been shifted one column to the right, it would have passed the mirror unaffected. We put some mirrors near places where signals might collide, so that (with its small neighbourhood) this rule can simulate an attractive collision – the signal paths will be displaced inward in a collision, rather than outward as in the BBM. See fig. 21. (If a signal arrives on

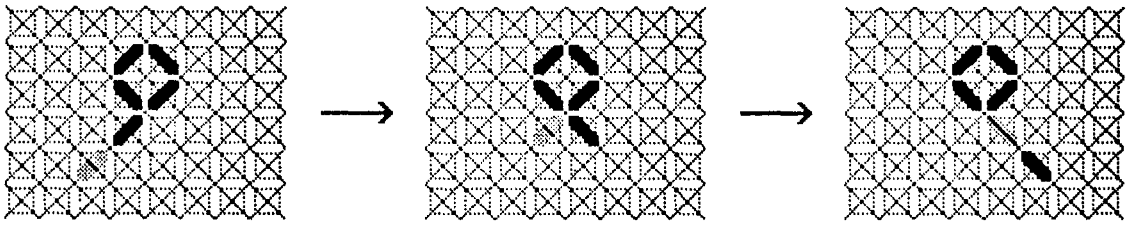


Fig. 20.

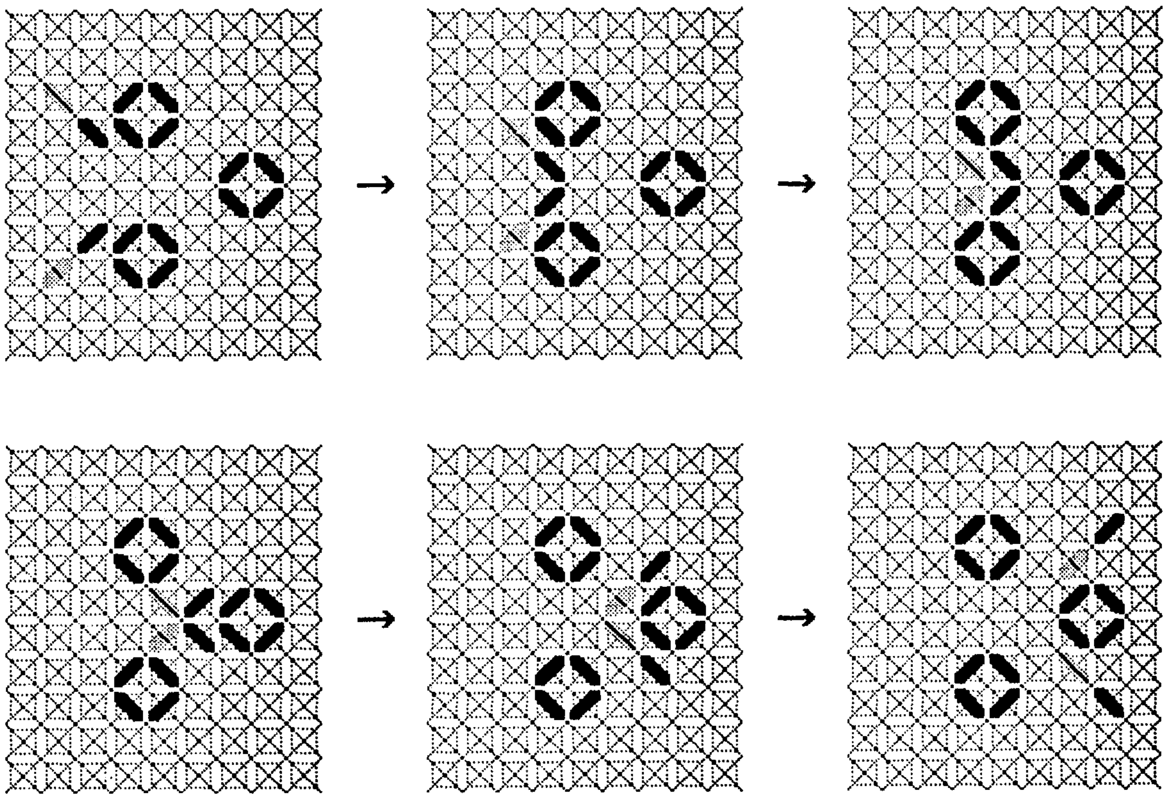
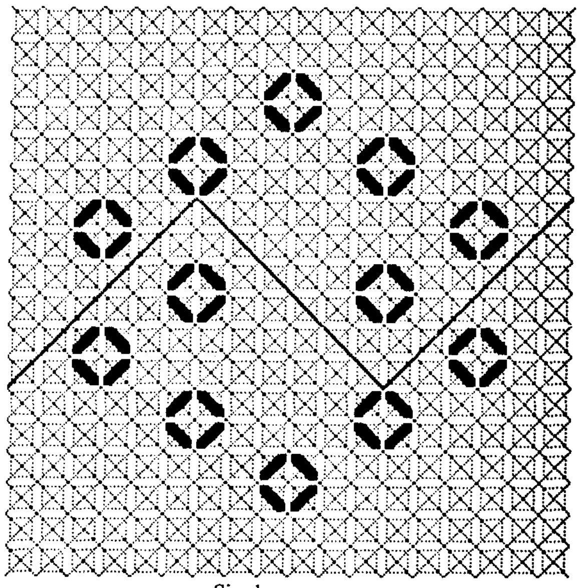


Fig. 21.

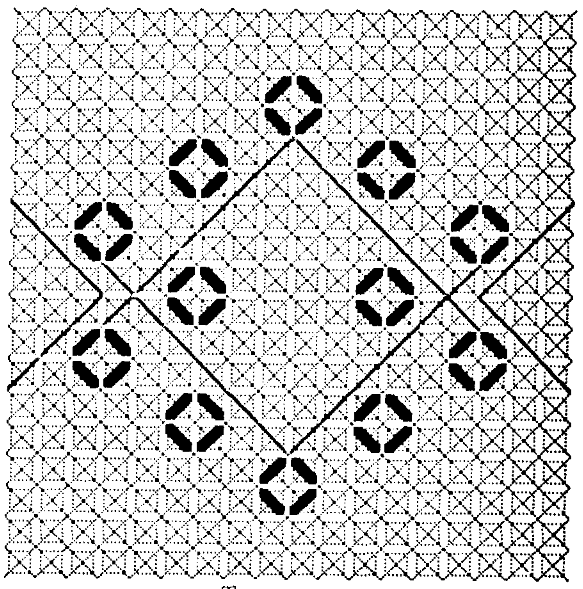
just one path, it goes through without any displacement). Two such gates, back to back, can be used to make signals cross over without affecting each other. See figs. 22 and 23.

Since all collisions occur without any delay along the plane of the collision, considerations of synchronization are very similar to those in the BBM. The proof of this automaton's universality is essentially the same as for the BBM.



Single one case

Fig. 22.



Two ones case

Fig. 23.

To give another example of a universal second-order RCA, we can begin with the BBMCA rule. If f_s is the global rule that applies to the solid blocking and changes an entire configuration into the next configuration, and similarly f_d applies to the dotted blocking, then we can describe the BBMCA evolution by

$$S_{t+1} + S_{t-1} = f(S_t), \tag{10}$$

where $S_{t+1} + S_{t-1}$ is taken to be the configuration obtained by taking the cell-by-cell sum (mod 2) of S_{t+1} and S_{t-1}, and $f(S_t) = f_s(S_t) + f_d(S_t)$ is also such a sum.

(10) can be rewritten in the form (4) with a 3×3 neighbourhood and a dependence on the parity of the center cell's position (parity of $x + y$). In a similar manner, *any* invertible CA rule can be written in the form (10), and in the form (4) if it is locally invertible.

References

[1] C.H. Bennet, "Logical reversibility of computation," IBM Journal of Research and Development **6** (1973) 85–91; "The Thermodynamics of Computation," Int. J. of Theo. Phys. **21** (1982) 905–940.

[2] E. Fredkin, private communication.

[3] E. Fredkin and T. Toffoli, "Conservative Logic," Int. J. of Theo. Phys. **21** (1982) 219–253.

[4] E. Berlekamp, J. Conway and R. Guy, "Winning Ways for your Mathematical Plays", vol. 2 (Academic Press, New York, 1982).

[5] M. Gardner, "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," Scientific American **223**:4 (1970) 120–123.

[6] R. Landauer, "Irreversibility and heat generation in the computing process," IBM Journal of Research and Development 5 (1961) 183–191.

[7] C. Shannon and W. Weaver, The Mathematical Theory of Communication (Univ. of Illinois Press, Illinois 1949).

[8] T. Toffoli, "Computation and construction universality of reversible cellular automata," Journal of computer systems science 15 (1977) 213–231.

[9] T. Toffoli, "CAM: A High-Performance Cellular-Automaton Machine," Physica 10D (1984) 195–204 (these proceedings).

[10] J. Von Neumann, Theory of Self-Reproducing Automata, (ed. and compiled by A.W. Burks) (Univ. Illinois Press, Illinois, 1966).