Logic circuits in a system of repelling particles

William M. Stevens

Department of Physics and Astronomy, Open University, Walton Hall, Milton Keynes, MK7 6AA, UK
william@stevens93.fsnet.co.uk,
WWW home page: http://www.srm.org.uk

Abstract. A model of a kinematic system consisting of moveable tiles in a two dimensional discrete space environment is presented. Neighbouring tiles repel one another, some tiles are fixed in place. A dual-rail logic gate is constructed in this system.

1 Introduction

This paper is motivated by a desire to understand how computing devices can be built from simple mechanical parts. An attempt is made to model the kinematic behaviour of a simple system without modelling any mechanism that gives rise to the kinematic behaviour in question. The model used can be classified as a 'kinematic automaton' and can be simulated using a cellular automaton.

This scheme arose during research into automatic construction and self-replication. Several researchers have proposed or built programmable constructing machines capable of automatically building a wide range of other machines under program control from a collection of prefabricated parts ([8], [5], [7]). If one is interested in simplifying the process of automatically constructing a complex device, then using a small range of component part types is advantageous because the complexity of the constructing device is likely to be lower than if a large range of part types were used.

Several abstract models of self-replicating systems based around programmable constructors have been devised ([8],[2],[11]), and at least one simple physical self-replicating system of this kind has been built ([12]). One problem that arises when considering how to make physical programmable self-replicators is that of devising a control system using the range of prefabricated parts that are available to the replicator. If some of the prefabricated parts have a built-in capability for processing digital information (as in [11]), or even a built-in computer (as in [12]), the problem is solvable, but the solution is not ideal because the complexity of such parts tends to be fairly high.

Some of the systems cited above are based around sets of parts that contain both mechanical elements and digital-processing elements, other systems are based on complex parts that combine both functions. It may be possible to reduce the range and/or the complexity of the set of parts required to build a self-replicating constructor by processing digital information using the mechanical interactions between parts and ultimately using a mechanical computer as the

control unit for a programmable constructor. This paper does not go that far, but does show how a logic gate can be made using simple mechanical interactions and a small range of part types.

This result is also of interest as a collision-based computing scheme. In [4], Fredkin and Toffoli showed that elastic collisions between idealised billiard balls (and fixed mirrors) obeying Newton's laws of motion can be used to implement boolean logic elements. Fredkin and Toffoli were interested in showing that reversible, conservative logic is physically possible. This paper is not concerned with conservative logic, but with logic elements made from simple parts that are technologically plausible. In [6] Margolus showed how to model Fredkin and Toffoli's system using a cellular automaton. Collision-based computing with a simple basis has also been demontrated in several other cellular automata environments ([9], [3]). While these environments are mathematically simple they are not designed to model physical behaviour.

2 A simple kinematic simulation environment

A two dimensional discrete space, discrete time simulation environment that supports moveable square tiles is studied in this paper. The behaviour of a tile is very simple: neighbouring tiles repel one another, but some tiles can be fixed in place. Figure 1 illustrates the behaviour of tiles, and should be read to mean that any occurance of the configuration to the left an arrow will evolve into the configuration to the right of the arrow after one time step. All other configurations remain unchanged.

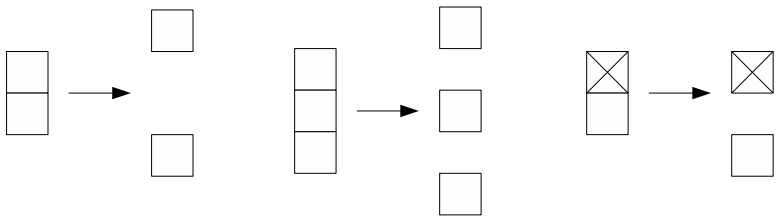


Fig. 1. Three rules that completely describe how tiles interact

This behaviour can be specified more formally using the set of cellular automaton rules given in figure 2.

These rules specify a cellular automaton with an eight cell neighbourhood and three states per cell. A cell can be empty, or it can contain a fixed tile, or it can contain a moveable tile. In figure 2 empty cells are denoted by a square with a dashed boundary, fixed tiles are denoted by a square with a cross in, moveable tiles are denoted by a square with a solid boundary. Cells with a dot in the centre can be in any of the three states. The rules are symmetrical, so if a

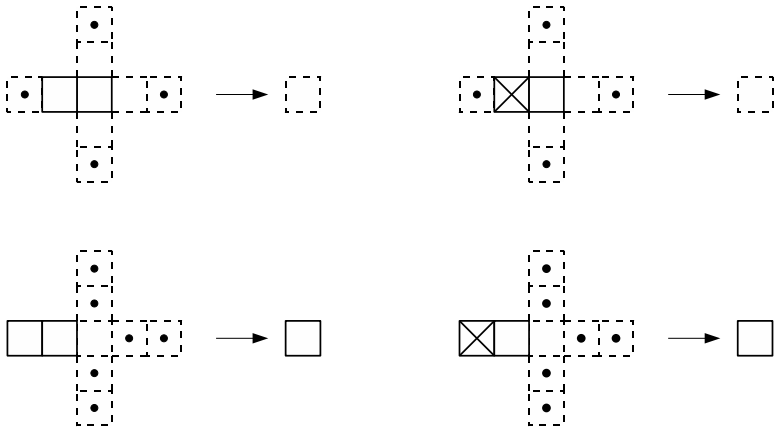


Fig. 2. Tile behaviour specified as a set of cellular automaton rules

configuration of tiles rotated through any multiple of 90 degrees matches a rule, the central cell changes to the state to the right of the arrow on the next time step. If a configuration matches none of these rules, the central cell remains in the same state.

These rules are sufficient for modelling all of the mechanisms described in this paper.

3 Some basic mechanisms

It is relatively straightforward to devise configurations of tiles that behave as 'one-shot' logic devices, in which arrangements of tiles compute a function once but cannot be re-used after the computation is finished. This paper shows how re-usable circuits can be constructed which are capable of carrying out one computation after another.

Figures 3 to 11 show nine basic mechanisms from which more complex mechanisms can be put together.

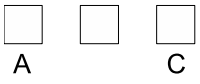


Fig. 3. Wire

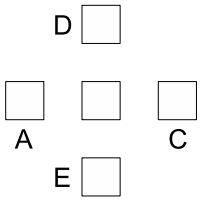


Fig. 4. Cross

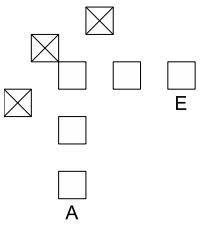


Fig. 5. Corner (Type 1)

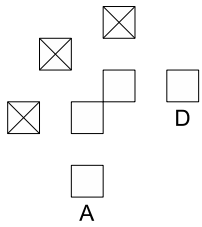


Fig. 6. Corner (Type 2)

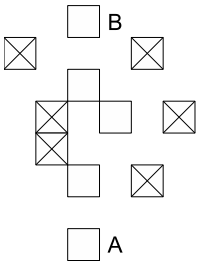


Fig. 7. Changer

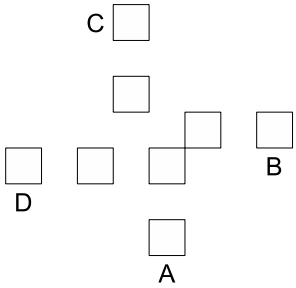


Fig. 8. Fan-out

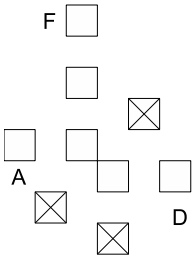


Fig. 9. Combine

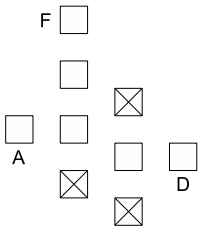


Fig. 10. Both

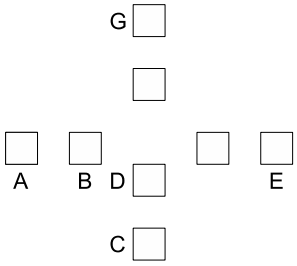


Fig. 11. Hold

The action of each of these mechanisms can be described formally. The notation used is intuitive and will be introduced as it is used. The letters n,e,s and w that appear in square brackets in equations denote the directions north (up the page), south (down the page), east (to the right of the page) and west (to the left of the page).

3.1 Wire

For the Wire in figure 3 we can write:

$$\begin{array}{l} \overset{1}{\nearrow} A[w] \\ A[e] \xrightarrow{2} C[e] \end{array} \tag{1}$$

To mean that the result of displacing A eastward by one unit at time t_A is that A will be displaced westward at $t_A + 1$, and C will be displaced eastward at $t_A + 2$. The centre tile in the Wire will also move, but will return to its original position. A Wire can thus be thought of as a path along which a signal can propagate, where the signal consists of a displacement of a tile away from its normal position in the path. Wires of any length can be made.

Note that a Wire also works in reverse, so we can also write:

$$\begin{array}{l} \overset{1}{\nearrow} C[e] \\ C[w] \xrightarrow{2} A[w] \end{array} \tag{2}$$

If both ends of a Wire are moved at once, the result is that two signals cancel each other out as follows:

$$\begin{array}{l} \overset{1}{\nearrow} A[w] \\ A[e],C[w] \xrightarrow{1} C[e] \end{array} \tag{3}$$

This 'cancelling out' behaviour is used extensively by the logic gate described in section 5.

3.2 Cross

For the Cross in figure 4 we can write:

$$\begin{array}{l} \overset{1}{\nearrow} A[w] \\ A[e] \xrightarrow{2} C[e] \end{array} \tag{4} \qquad \begin{array}{l} \overset{1}{\nearrow} C[e] \\ C[w] \xrightarrow{2} A[w] \end{array} \tag{5}$$

$$\begin{array}{l} D[n] \\ D[s] \xrightarrow{2} E[s] \end{array} \quad (6) \qquad \begin{array}{l} E[s] \\ E[n] \xrightarrow{2} D[n] \end{array} \quad (7)$$

$$\begin{array}{l} A[w] \\ A[e],C[w] \xrightarrow{1} C[e] \end{array} \quad (8) \qquad \begin{array}{l} D[n] \\ D[s],E[n] \xrightarrow{1} E[s] \end{array} \quad (9)$$

Note that the Cross mechanism misbehaves when $(A[e] \vee C[w]) \wedge (D[s] \vee E[n])$ at any time t, so any circuit that uses the Cross mechanism must avoid this situation.

3.3 Corner (Type 1)

For the Type 1 Corner in figure 5 we can write:

$$\begin{array}{l} A[s] \\ A[n] \xrightarrow{4} E[e] \end{array} \quad (10) \qquad \begin{array}{l} E[e] \\ E[w] \xrightarrow{4} A[s] \end{array} \quad (11)$$

3.4 Corner (Type 2)

For the Type 2 Corner in figure 6 we can write:

$$\begin{array}{l} A[s] \\ A[n] \xrightarrow{3} D[e] \end{array} \quad (12) \qquad \begin{array}{l} D[e] \\ D[w] \xrightarrow{3} A[s] \end{array} \quad (13)$$

Note that both types of Corner can propagate signals in either direction. If we only need a Corner to work in one direction then one of its fixed tiles can be removed.

3.5 Changer

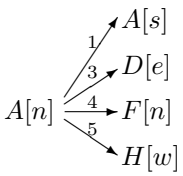
For the Changer in figure 7 we can write:

$$\begin{array}{l} A[s] \\ A[n] \xrightarrow{7} B[n] \end{array} \quad (14)$$

The Changer is so-called because it alters the spacing of the tiles in a signal path. This is often necessary when joining mechanisms together.

3.6 Fanout

For the Fanout mechanism in figure 8 we can write:

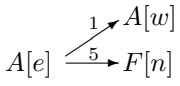


(15)

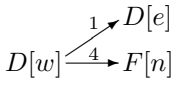
If we need fewer than 3 outputs, any of the output paths in the Fanout mechanism can be replaced with a single fixed tile.

3.7 Combine

For the Combine mechanism in figure 9 we can write:

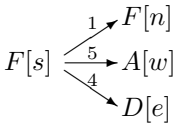


(16)



(17)

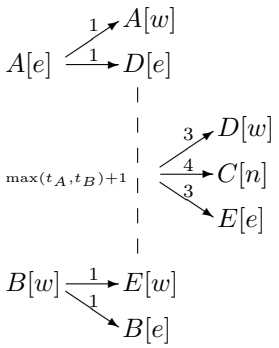
The Combine mechanism also has useful behaviour when driven from tile F:



(18)

3.8 Both

The Both mechanism will produce an output only after both of its inputs have been stimulated:



(19)

In equation 19 the dashed line with arrows leading to subsequent events indicates that the occurrance of two events (at time $\max(t_A, t_B) + 1$) causes the subsequent events.

Because tiles D and E return to their original positions during the operation of the Both mechanism its behaviour can be described more concisely:

$$\begin{array}{l} A[e] \xrightarrow{1} A[w] \\ \quad | \\ \quad |_{\max(t_A,t_B)} \\ \quad | \qquad \xrightarrow{5} C[n] \\ \quad | \\ \quad | \\ \quad | \\ B[w] \xrightarrow{1} B[e] \end{array} \tag{20}$$

3.9 Hold

The Hold mechanism in figure 11 consists of four paths meeting at a junction, and is used as follows. At time t_A tile A may or may not be displaced eastward. At time t_E tile E may or may not be displaced westward. So that at time $t_m = \max(t_A, t_E) + 1$ the Hold mechanism may be in one of the four possible states shown in figures 11 to 14. At time $t_C \geq t_m$ tile C is displaced northward, and the response of the mechanism depends upon which of the four states it is in. Let us call the arrangements shown in figures 11, 12, 13 and 14 H_0, H_1, H_2 and H_3 respectively.

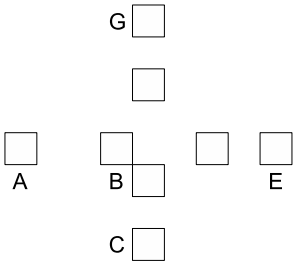


Fig. 12. H_1

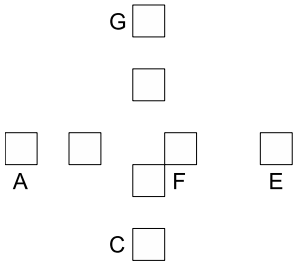


Fig. 13. H_2

Of H_0 we can say:

$$\begin{array}{l} \qquad\quad \overset{1}{\nearrow} C[s] \\ C[n] \xrightarrow{1} D[n] \end{array} \tag{21}$$

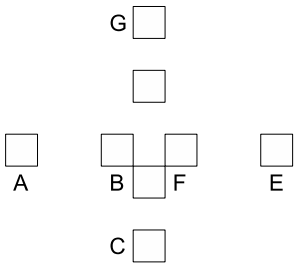


Fig. 14. H_3

Of H_1 we can say:

$$C[n] \xrightarrow{1} C[s] \quad C[n] \xrightarrow{2} B[w] \quad C[n] \xrightarrow{4} E[e] \tag{22}$$

Of H_2 we can say:

$$C[n] \xrightarrow{1} C[s] \quad C[n] \xrightarrow{2} F[e] \quad C[n] \xrightarrow{4} A[w] \tag{23}$$

And of H_3 we can say:

$$C[n] \xrightarrow{1} C[s] \quad C[n] \xrightarrow{2} B[w] \quad C[n] \xrightarrow{2} F[e] \tag{24}$$

What we have in effect is a mechanism that allows us to collide two signals (entering at A and E) together without having to worry about the relative timing of the two signals, because the collision only takes place when a signal is applied at C. When the Hold mechanism is used in a circuit, any signals emerging from A or E after the collision will propagate away from the Hold mechanism, leaving it in the state shown in figure 15, which we call H_4. We can return the mechanism to its original state H_0 by applying a signal at G.

For H_4:

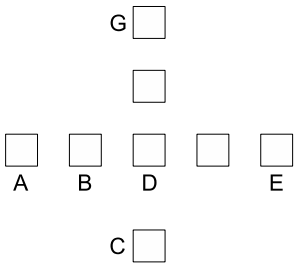


Fig. 15. H_4

$$G[s] \xrightarrow{1} G[n] \qquad G[s] \xrightarrow{2} D[s] \tag{25}$$

Table 1 shows the logical operations that are effected by the collision of two signal paths in a Hold mechanism.

Inputs before t_C	Outputs after t_C
Neither A nor E	Neither A nor E
A but not E	E but not A
E but not A	A but not E
Both A and E	Neither A nor E

Table 1. A collision between two signal paths

The fact that a signal will only be output at E if a signal is input at A but not at E before time t_C forms the basis of the logic gate described in section 5.

4 Circuits

Circuits can be made by connecting mechanisms together, and in the next section a dual-rail logic gate is made using the nine mechanisms described in the previous section. Before doing this, it is necessary to show how to describe the behaviour of two mechanisms joined to one another in such a way that a signal emerging from one will enter another.

Figure 16 shows a Wire and a Combine mechanism that have tile B in common. We saw earlier how to describe the behaviour of each of these mechanisms individually, and using this knowledge we can write:

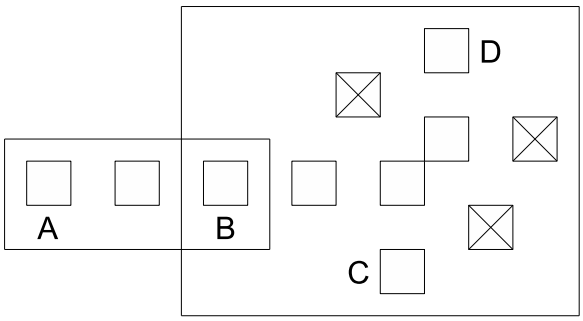


Fig. 16. Two mechanisms joined via tile B

$$\begin{array}{ccccc} & \overset{1}{\nearrow} & A[w] & \overset{1}{\nearrow} & B[w] \\ A[e] & \xrightarrow{2} & B[e] & \xrightarrow{5} & C[s] \\ & & & \underset{4}{\searrow} & D[n] \end{array} \tag{26}$$

In words, this means that a displacement of A eastward by one unit at t_A causes A to be displaced westward at $t_A + 1$ and B to be displaced eastward at $t_A + 2$. The displacement of B subsequently causes B to be displaced westward at $t_A + 2 + 1$, C to be displaced southward at $t_A + 2 + 5$ and D to be displaced northward at $t_A + 2 + 4$.

Because there is no net movement of B, we can shorten 26 to:

$$\begin{array}{ccc} & \overset{1}{\nearrow} & A[w] \\ A[e] & \xrightarrow{7} & C[s] \\ & \underset{6}{\searrow} & D[n] \end{array} \tag{27}$$

5 A dual-rail logic gate

In subsection 3.9 we saw how the Hold mechanism effects a logical operation. In order to use this logical operation as the basis for a logic gate, additional circuitry is needed.

The A and E inputs to the Hold mechanism also serve as outputs after a signal has been applied at C. To enable us to extract a signal returning along a path which is also used as an input we can use the mechanism shown in figure 17, called a 'uni-directional gate with tap'. An analysis of this mechanism is given in equations 28 and 29.

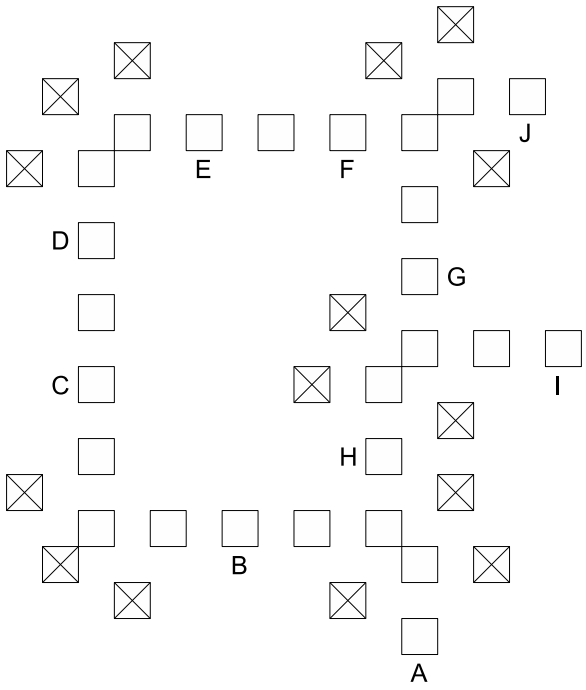


Fig. 17. Uni-directional gate with tap

$$\begin{array}{ccccccccccccc} & \overset{1}{\nearrow} & A[s] & & \overset{1}{\nearrow} & B[e] & & \overset{1}{\nearrow} & C[s] & & \overset{1}{\nearrow} & D[s] & & \overset{1}{\nearrow} & E[w] & & \overset{1}{\nearrow} & F[w] & & \overset{1}{\nearrow} & G[n] \\ A[n] & \underset{4}{\rightarrow} & B[w] & \underset{6}{\rightarrow} & C[n] & & \underset{2}{\rightarrow} & D[n] & & \underset{3}{\rightarrow} & E[e] & & \underset{2}{\rightarrow} & F[e] & & \underset{5}{\rightarrow} & G[s] & & \underset{5}{\rightarrow} & I[e] \end{array} \tag{28}$$

$$\begin{array}{ccccccccccc} & \overset{1}{\nearrow} & I[e] & & \overset{1}{\nearrow} & G[s] & & \overset{1}{\nearrow} & F[e] & & \overset{1}{\nearrow} & E[e] \\ I[w] & \underset{5}{\rightarrow} & G[n] & \underset{5}{\rightarrow} & F[w] & & \underset{2}{\rightarrow} & E[w] & & \underset{3}{\rightarrow} & D[s] \\ & & & \underset{4}{\searrow} & J[e] & & & & & & \vdots \\ & \underset{4}{\searrow} & & & & & & & & \overset{1}{\nearrow} & D[n] \\ & & & & & & & & t_I+15 & \underset{1}{\rightarrow} & C[s] \\ & & & & & & & & \vdots & & \\ & & & \overset{1}{\nearrow} & H[n] & & \overset{1}{\nearrow} & B[e] & \vdots & & \\ H[s] & \underset{5}{\rightarrow} & B[w] & & & & \underset{6}{\rightarrow} & & C[n] & & \end{array} \tag{29}$$

In equation 29 (as in equation 19) the dashed line with arrows leading to subsequent events indicates that the occurrance of two events (at time $t_I + 15$ in this case) causes the subsequent behaviour (in this case, the behaviour of a wire when both ends are displaced at once).

By observing which tiles return to their original positions during the course of the operation of this mechanism, we can shorten these equations to:

$$\begin{array}{l} \overset{1}{\nearrow} A[s] \\ A[n] \xrightarrow{27} I[e] \end{array} \tag{30}$$

$$\begin{array}{l} \overset{1}{\nearrow} I[e] \\ I[w] \xrightarrow{9} J[e] \end{array} \tag{31}$$

Thus, by applying a signal to A we can inject a signal onto a path connected to I. If a signal enters the mechanism at I, it will emerge at J.

Note that the number of tiles used in this mechanism could be reduced by shortening some of the paths. However, if we were to do this then we would not be able to analyse the mechanism in terms of the nine basic mechanisms described previously. To simplify the analysis and to make it clear which of the basic mechanisms we are using, we will use longer paths than are necessary.

A derivative of the mechanism in figure 17 is shown in figure 18. This is called a 'uni-directional gate'. The behaviour of the uni-directional gate is given in equations 32 and 33.

$$\begin{array}{l} \overset{1}{\nearrow} A[s] \\ A[n] \xrightarrow{27} I[e] \end{array} \tag{32}$$

$$I[w] \xrightarrow{1} I[e] \tag{33}$$

Thus, the uni-directional gate will permit a signal to travel from A to I, but not in the reverse direction.

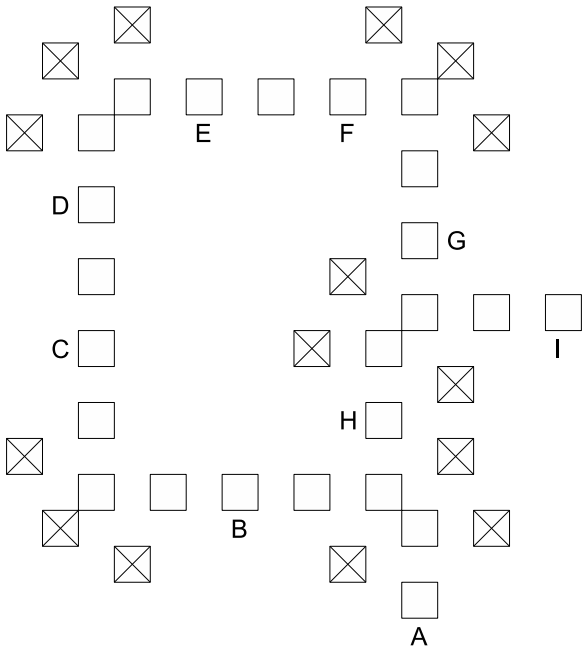


Fig. 18. Uni-directional gate

Recall that in dual-rail or 1-of-2 logic, every logical value X is represented by a pair of binary values X_0 and X_1 so that $X \leftrightarrow (X_0 = 0 \wedge X_1 = 1)$ and $\bar{X} \leftrightarrow (X_0 = 1 \wedge X_1 = 0)$. We can implement a dual-rail logic scheme by using two signal paths corresponding to X_0 and X_1 in such a way that the passage of a signal along one path represents logic 0, and the passage of a signal along the other path represents logic 1. For a dual-rail logic gate with two logical inputs A and B (and therefore four signal path inputs A_0, A_1, B_0 and B_1), it is possible to detect when both logical inputs have been received using the mechanism shown in figure 19.

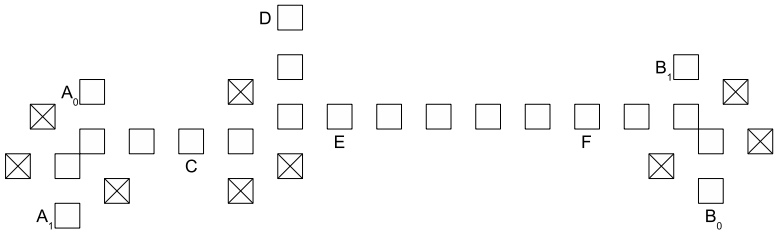
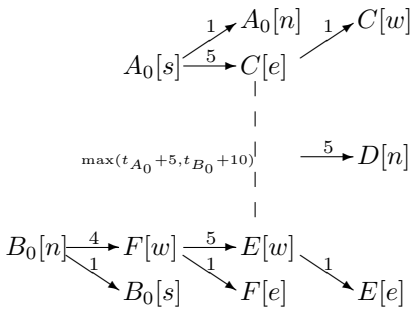


Fig. 19. Dual-rail input detection mechanism

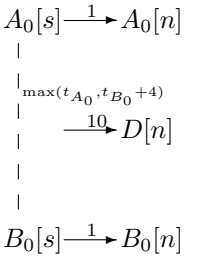
There are 4 possible cases to analyse for this mechanism :

A	B	A_0	A_1	B_0	B_1	Equation
False	False	1	0	1	0	35
False	True	1	0	0	1	36
True	False	0	1	1	0	37
True	True	0	1	0	1	38

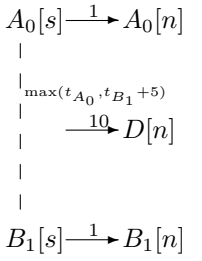
A complete analysis for the first case in this table is given in equation 34, which can be shortened to equation 35. Complete analyses for the other three cases are omitted for the sake of brevity since they follow a similar pattern to equation 34.



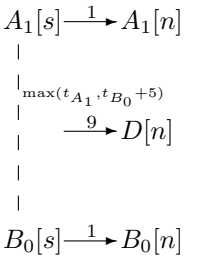
(34)



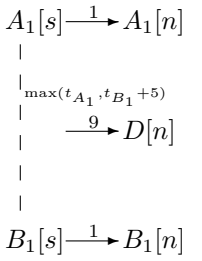
(35)



(36)



(37)



(38)

Equations 35 to 38 show that a signal will emerge at D for for every possible combination of logical values that A and B can take.

When describing the Hold mechanism shown in figure 11 we noted that it will be used by firstly applying signals at A or E (or both, or neither), then applying a signal at C, and afterwards applying a signal at D to reset the mechanism. The augmented hold mechanism shown in figure 20 allows us to dispense with having to apply a reset signal by deriving a reset signal from the signal at C.

The net behaviour of the augmented hold mechanism is given in equations 39 to 42 for the four possible cases that result if signals are applied at A or B or both or neither.

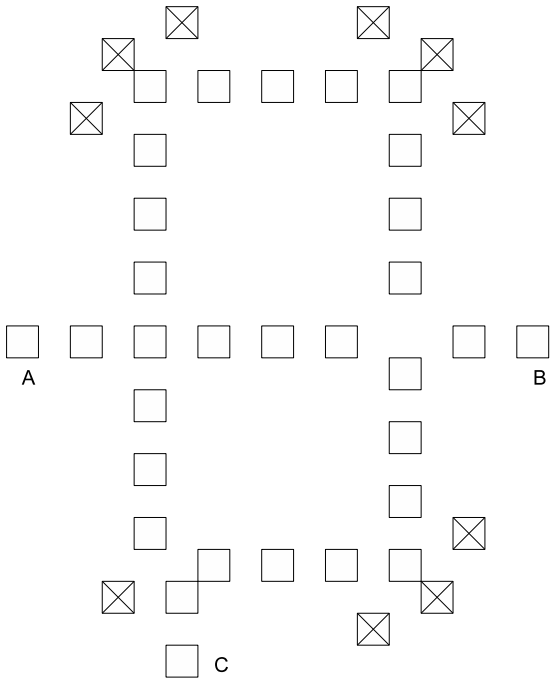


Fig. 20. Augmented hold mechanism

$$\begin{array}{l} A[e] \xrightarrow{1} A[w] \\ \quad\vdots \\ \quad\vdots^{\max(t_A,t_C+5)} \\ \quad\vdots \xrightarrow{8} B[e] \\ \quad\vdots \\ C[n] \xrightarrow{1} C[s] \end{array} \qquad (39)$$

$$\begin{array}{l} B[w] \xrightarrow{1} B[e] \\ \quad\vdots \\ \quad\vdots^{\max(t_B,t_C+5)} \\ \quad\vdots \xrightarrow{8} A[w] \\ \quad\vdots \\ C[n] \xrightarrow{1} C[s] \end{array} \qquad (40)$$

$$\begin{array}{l} A[w] \xrightarrow{1} A[e] \\ B[w] \xrightarrow{1} B[e] \\ C[n] \xrightarrow{1} C[s] \end{array} \qquad (41)$$

$$C[n] \xrightarrow{1} C[s] \qquad (42)$$

Equation 41 holds so long as $t_C \geq \max(t_A - 5, t_B - 9)$.

Using a uni-directional gate with tap, a uni-directional gate, an augmented hold mechanism, a dual-rail input detection mechanism, two Combine mechanisms, a Changer and some connecting wires we can make the mechanism shown in figure 21. This mechanism feeds signals derived from A_0 and B_1 into an augmented hold mechanism, where a collision will take place once the dual-rail input detect mechanism indicates that all required inputs have been received. After the collision a signal will emerge at O if a signal was applied at A_0 but not at B_1. A signal derived from the output of the dual-rail input detect mechanism will emerge at P regardless of the logical values of A and B.

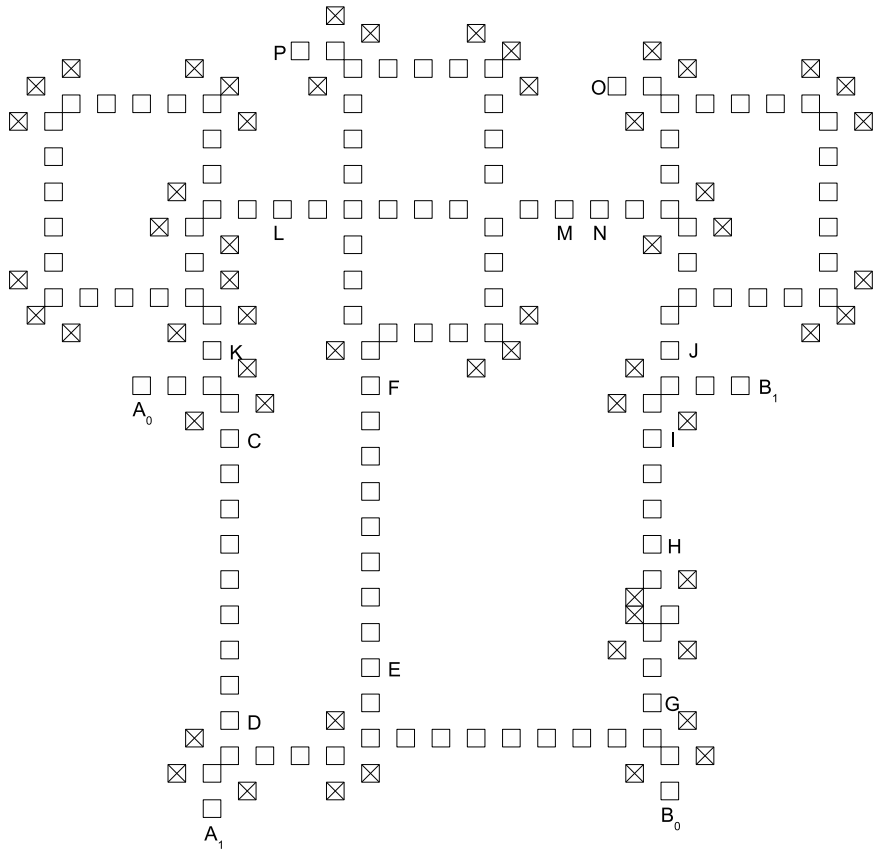
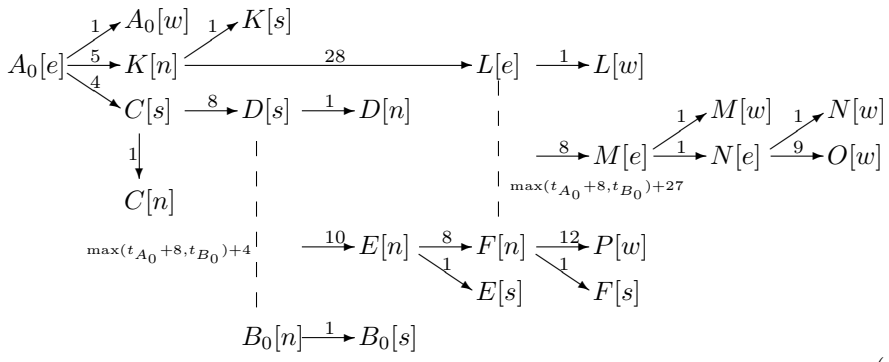


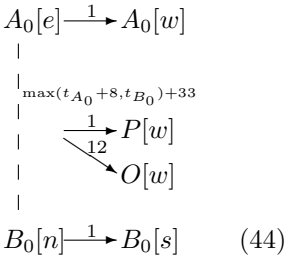
Fig. 21. Logic gate - Stage 1

Equation 43 shows an analysis of the case for $A_0[e]$ at time t_{A_0} and $B_0[n]$ at time t_{B_0}.

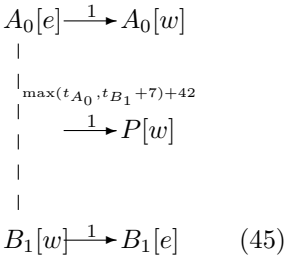


(43)

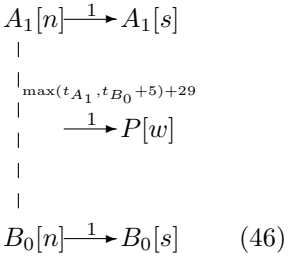
This can be simplified to equation 44. Equations for the other three cases are shown in 45 to 47. (Complete analyses for these are not given since they follow a similar pattern to 43).



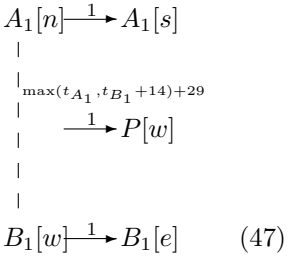
(44)



(45)



(46)



(47)

Thus in all four cases, a signal will emerge at P.

In the case shown in equation 44 (where signals are applied at A_0 and B_0), a signal will emerge at P at time $t_P = \max(t_{A_0} + 8, t_{B_0}) + 34$ and another signal will emerge at O at time $t_O = \max(t_{A_0} + 8, t_{B_0}) + 45 = t_P + 11$.

The mechanism shown in figure 22 can be used to derive a signal from O to use as the Q_1 output of a dual-rail logic gate, and to derive a signal from P and O to use as the Q_0 output of a dual-rail logic gate.

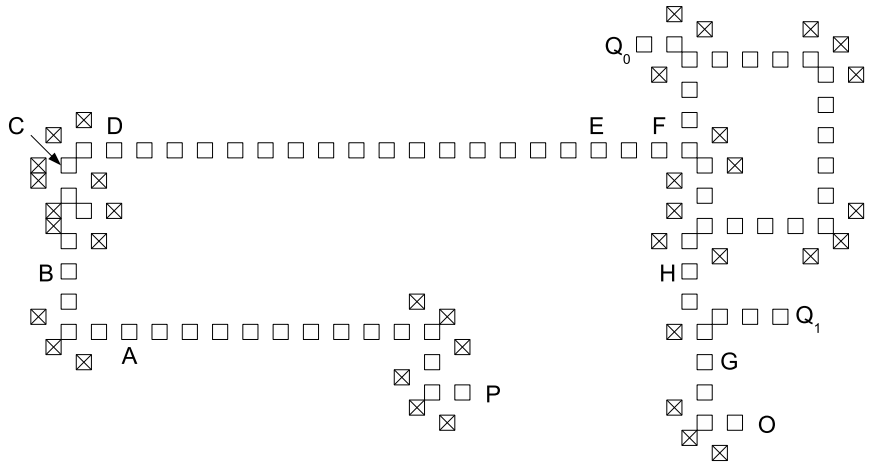


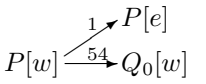
Fig. 22. Logic gate - Stage 2

An analysis of this mechanism is given in equations 48 and 49.

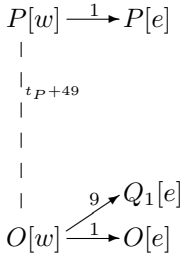
$$\begin{array}{ccccccccccc} & \overset{1}{\nearrow} P[e] & & \overset{1}{\nearrow} A[e] & & \overset{1}{\nearrow} B[s] & & \overset{3}{\nearrow} C[s] & & \overset{1}{\nearrow} D[w] & & \overset{1}{\nearrow} F[w] \\ P[w] & \xrightarrow[17]{} & A[w] & \xrightarrow[6]{} & B[n] & \xrightarrow[1]{} & C[n] & \xrightarrow[2]{} & D[e] & \xrightarrow[17]{} & F[e] & \xrightarrow[9]{} & Q_0[w] \end{array} \tag{48}$$

$$\begin{array}{ccccccccccc} & \overset{1}{\nearrow} P[e] & & \overset{1}{\nearrow} A[e] & & \overset{1}{\nearrow} B[s] & & \overset{1}{\nearrow} C[s] & & \overset{1}{\nearrow} D[w] & & & \\ P[w] & \xrightarrow[17]{} & A[w] & \xrightarrow[6]{} & B[n] & \xrightarrow[7]{} & C[n] & \xrightarrow[2]{} & D[e] & \xrightarrow[17]{} & E[e] & & \\ & & & & & & & & & & \vdots & \overset{}{\nearrow} & E[w] \\ & & & & & & & & & & & \overset{1}{\searrow} & F[e] \\ & & & & & & & \overset{4}{\nearrow} & Q_1[e] & & \vdots & & \\ O[w] & \xrightarrow[5]{1} & G[n] & & & & & \xrightarrow[1]{4} & H[n] & \xrightarrow[1]{27} & F[w] & & \\ & \searrow & O[e] & & & & & \searrow & G[s] & \searrow & H[s] & & \end{array} \tag{49}$$

These equations can be simplified to 50 and 51 and used in conjuntion with 44 to 47 to deduce the overall behaviour of the logic gate shown in figure 23.



(50)



(51)

A	B	Inputs to gate	Outputs from gate	Q
False	False	$A_0[e]$ at t_{A_0} , $B_0[n]$ at t_{B_0}	$Q_1[e]$ at $\max(t_{A_0}+8, t_{B_0})+92$	True
False	True	$A_0[e]$ at t_{A_0} , $B_1[w]$ at t_{B_1}	$Q_0[e]$ at $\max(t_{A_0}, t_{B_1}+7)+97$	False
True	False	$A_1[n]$ at t_{A_1} , $B_0[n]$ at t_{B_0}	$Q_0[e]$ at $\max(t_{A_1}, t_{B_0}+5)+84$	False
True	True	$A_1[n]$ at t_{A_1} , $B_1[w]$ at t_{B_1}	$Q_0[e]$ at $\max(t_{A_1}, t_{B_1}+14)+84$	False

Table 2. Logical behaviour of the gate in figure 23

Thus, the overall behaviour of our dual-rail logic gate is described by table 2. From this, it can be seen the the logic gate in figure 23 is a dual-rail implementation of a boolean NOR gate.

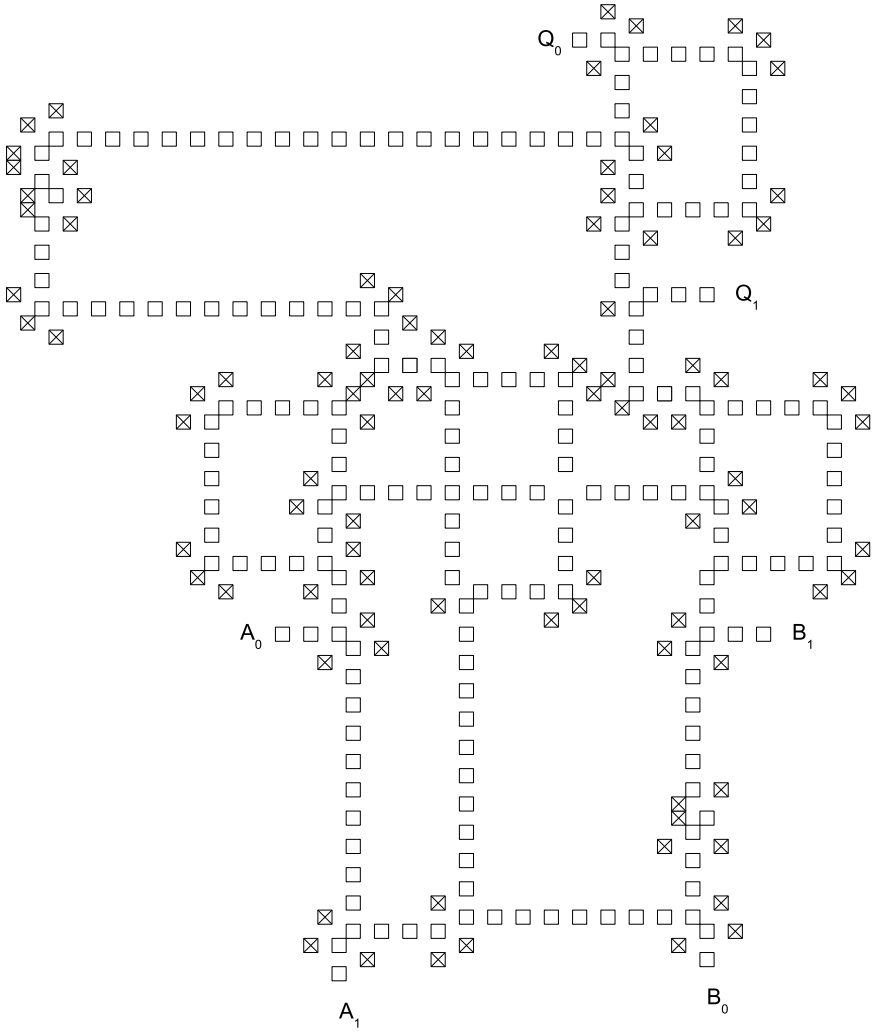


Fig. 23. A dual-rail NOR gate

6 Conclusion

The number of tiles in figure 23 could be reduced by shortening the paths between mechanisms and by reducing the size of some mechanisms which were deliberately kept larger than necessary in order to clarify their structure. Unused fixed tiles could also be removed at some corners and in some Combine mechanisms.

The part of the logic gate that performs the logic operation is the augmented hold mechanism. If we placed constraints on signal timing at the inputs to the gate, we could do without this mechanism and replace it with a wire, resulting in a simpler gate. However, if we were to do this and then attempt to connect several logic gates together to make a circuit we would have to introduce delays between one gate and another in order to meet timing constraints.

Many tiles in the logic gate are involved with separating signals travelling in one direction along a path from signals travelling in the opposite direction. Note that in Fredkin and Toffoli's Billiard Ball model and in some logic schemes based on glider collisions in two and three dimensionsal cellular automata (for example [10]) signal separation of this kind is not necessary because in these environments particles can be given a velocity component perpendicular to the collision axis, so that the results from a collision automatically end up in a different location from the 'inputs' to the collision.

One aim of this work was to find a simple and technologically plausible basis for computating using a small range of simple kinematical part types. To assess whether the system described meets this aim, further work is needed. Are there any physical systems of repelling particles in which the system described can be implemented?

One problem that may need to be addressed in a physical system with classical behaviour is that of emulating a synchronously-updating discrete grid in an asynchronous continuous system. One approach may be to arrange things so that the substrate on which particles move lies closely parallel to a regular array of attractors that can be switched on and off periodically, and to which particles are attracted so strongly that the repulsion between neighbouring particles can be overcome. When the attractors are switched on, the particles will align themselves with the regular array. When the attractors are switched off, the particles can interact and move.

Another approach may be to adapt the system described here so that global synronization of the system is not required. Adachi et. al. ([1]) have shown that some asynchronous cellular automata are capable of supporting universal computation. Further work is needed to determine whether the CA rules specified in figure 2 support universal computation if used in an ACA model.

Such speculations cannot proceed far without deeper research into the physics of repelling particles in various different physical environments.

Software to simulate the system described in this paper can be obtained at http://www.srm.org.uk

References

1. Adachi, S., Peper, F., Lee, Jia.: Computation by Asynchronously Updating Cellular Automata. J. Stat. Phys. **114(1/2)** (2004) 261–289
2. Codd, E.F.: Cellular Automata. Academic Press, New York (1968)
3. Cook, M.: Universality in Elementary Cellular Automata. Complex Systems **15** (2004) 1–40
4. Fredkin, E., Toffoli, T.: Conservative Logic. J. Theo. Phys. (1982) 219–253
5. Freitas, R., Gilbreath, W. P.: Advanced Automation for Space Missions. Section 5.6.4: Robot Replication Feasability Demonstration. NASA Conference Publication CP-2255 (1982) 253–257
6. Margolus, N.: Physics-Like Models of Computation. Physica D **10** 1984 81-95
7. Moses, M.: A Physical Prototype of a Self-Replicating Universal Constructor. Masters Thesis, Department of Mechanical Engineering, University of New Mexico (2001). http://www.home.earthlink.net/˜mmoses152/SelfRep.doc
8. Von Neumann, F.: Theory of Self-Reproducing Automata. Edited and completed by A.W. Burks. University of Illinois Press, Urbana Illinois (1966) 81–82
9. Rendell, P.: Turing Universality of the Game of Life. Collision-Based Computing (2002) 513–539
10. Rennard, J.: Implementation of Logical Functions in the Game of Life. Collision-Based Computing (2002) 491–512
11. Stevens, W.M.: A programmable constructor in a kinematic environment. Poster presentation at Micro and Nanotechnology 2005 conference. http://www.srm.org.uk/papers/CBlocks3D_programmable_constructor.pdf
12. Zykov, V., Mytilinaios, E., Adams, B., Lipson, H.: Self-Reproducing Machines. Nature **435(7038)** (2005) 163–164.