Int. J. of Unconventional Computing, Vol. X, pp. 1–25
Reprints available directly from the publisher
Photocopying permitted by license only

Demonstration of the Universality of a New Cellular Automaton

E. SAPIN[1], O. BAILLEUX[1], J.-J. CHABRIER[1] AND P. COLLET[2]

[1]*Université de Bourgogne, 9 avenue A. Savary, B.P. 47870, 21078 Dijon Cedex, France*
E-mail: {olivier.bailleux, jjchab}@u-bourgogne.fr, emmanuelsapin@hotmail.com
[2]*Laboratoire d'Informatique du Littoral, Université du Littoral, Calais, France*
E-mail: pierre.collet@univ-littoral.fr

Received: September 20, 2005. In Final Form: September 30, 2005.

The paper deals with classification and search for universality of cellular automata — one of the central issues of cellular automata theory. The problem of search for universality was posed by Wolfram in *Twenty Problems in the Theory of Cellular Automata* — "How common are computational universality and undecidability [are] in cellular automata?". We demonstrate how universal automata can be designed and found using evolutionary algorithms. Namely, we show how to evolved a new automaton (called R) that can implement the Game of Life (which is universal in the Turing sense).

All elements of the evolutionary algorithms that were used to find R are provided for replicability, as well as the analytical description of representing a cell of the Game of Life in cellular automaton R.

Keywords: Cellular Automata, Computational Universality, Evolutionary Algorithms, Game of Life, Rule R, Glider Gun, Eater, NAND Gate.

1 INTRODUCTION

Cellular automata are discrete systems [1] in which a population of cells evolves from generation to generation on the basis of local transition rules. They can simulate simplified "forms of life"[2, 3] or physical systems with discrete time, space, and local interactions [4–6].

We choose to study a famous problem, related to the emergence of computation in complex systems with simple local behaviour[7]:

– Where are edges of computational universality ?
– How common is computational universality ?

The last question was posed by Wolfram in [8]: “How common are computational universality and undecidability in cellular automata?” This paper gives elements of answer to this problem as it describes how a universal cellular automaton was sought *and found* using evolutionary algorithms.

In [9], Conway showed the universality of a special automaton called *the Game of Life* (hereafter called Life). This proof is quite complex, and uses the presence of *gliders*, i.e. patterns which, when evolving alone, periodically recover their original shape after some shift in space. This paper presents how an evolutionary algorithm allowed to find R, another universal cellular automaton than Life. Rather than going through a complete demonstration of universality, we show how NAND gate is simulated by R (therefore showing the logic universality of R).

Then, one cell of Life is simulated and a way to tile a two dimensional space with identical and adjacent simulations of Life cells is given. The conclusion is that if Life is computationally universal, then R is computationally universal too, because R can implement Life.

Section 2 describes previous work. Section 3 describes the search process, that is developed in Sections 4 and 5. Section 6 shows a proof of universality of the automaton R, and Section 7 summarizes the presented results and discusses directions for future research.

2 PREVIOUS WORK

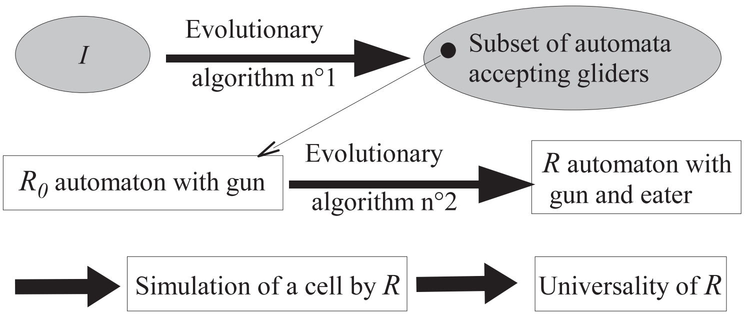
2.1 Universal automata

In [10], Wolfram studied $\mathcal{E}$, a space of two-state 2D automata with a transition rule that takes into account the eight neighbours of a cell in order to determine its state at the next generation. Within this space, Wolfram studied more precisely the subset of *isotropic* automata. In isotropic automata, if two cells have the same neighbourhoods by rotations and symmetries, then these two cells have the same state at the next generation.

There are 102 neighbourhoods that are different by rotations and symmetries, meaning that there are 2^{102} different automata in the subset $\mathcal{I} \subset \mathcal{E}$ of $\mathcal{I}$sotropic automata.

In 1970, Conway discovered a special automaton of $\mathcal{I}$ (that he called the *Game of Life*) that was later popularised by Gardner in [2]. In [9], Conway, Berlekamp, and Guy show that Life can implement any function calculable by a Turing machine. Their proof of the universality of Life uses *gliders*, *glider guns*, and *eaters*. A glider gun is a pattern that emits a stream of gliders (used to carry information) and an eater absorbs gliders.

It is possible to combine glider guns and eaters in order to simulate logic gates and circuits. In [11], Rendell gives an explicit proof of the universality of Life by showing a direct simulation of counter machines.



I
Evolutionary
algorithm n°1
Subset of automata accepting gliders
R_0 automaton with gun
Evolutionary
algorithm n°2
R automaton with gun and eater
Simulation of a cell by R
Universality of R

FIGURE 1
Looking for a computationally universal automata.

However, until now, Life was the only known dynamic universal automaton of $\mathcal{I}$. Some automata, like Banks's 2D 2-state cellular automaton [12] are not dynamically universal because theirs wires are fixed. Margolus's billard-ball model cellular automaton [13] is universal but its transition rules takes into account the parity of the number of generations, meaning that it is not in $\mathcal{I}$. Then, universal automata with more than two states [14, 15] or more than two dimensions [16] are not in $\mathcal{I}$ either.

2.2 Evolutionary algorithm

In order to search for another universal automata than Life, we used an evolutionary algorithm [17], that incorporates aspects of natural selection or survival of the fittest. It maintains a population of structures (usually initially generated at random) that evolves according to rules of selection, recombination, mutation, and survival referred to as genetic operators. A shared "environment" is used to determine the fitness or performance of each individual in the population. The fittest individuals are more likely to be selected for reproduction through recombination and mutation, in order to obtain potentially superior ones.

3 RATIONALE

Figure 1 shows how new universal automata are discovered with Evolutionary Algorithms. Our goal is to find new universal automata in the set $\mathcal{I}$ of isotropic automata.

In [9], Conway *et al.* use special patterns (gliders and eaters) to demonstrate the universality of Life. In [18], Heudin also studies gliders in order to search for the universality of automata. Following these observations, we decided that in order to find another computationally universal automaton than Life, the first step was to look for automata in $\mathcal{I}$ accepting gliders.

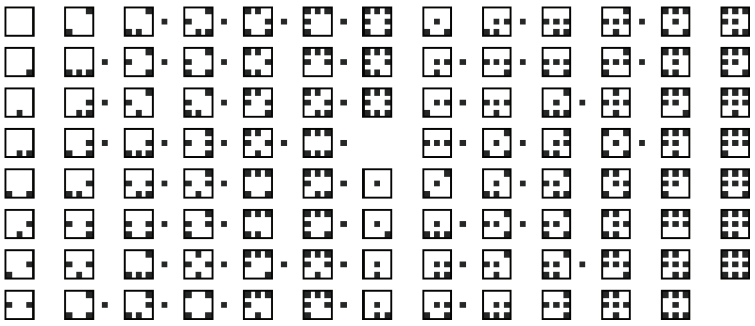


FIGURE 2
The squares are the 102 neighbourhoods describing an automaton of $\mathcal{I}$. A black cell on the right of the neighbourhood indicates a future central cell. The R_0 automaton, can then be represented in binary by: 000 00000011 10001110 11111111 11111100 10010111 11111000 (for the neighbourhoods without a central cell, by reading top-down and left-right) and 000 00111011 11101110 00010001 01101000 00000000 00000000 (for the neighbourhoods with a central cell), a 0 or a 1 representing the state of the future central cell in the next generation. This can be represented in a shorter way in hexadecimal by 00 03 8E FF FC 97 F8 (for the empty central cells neighbourhoods, with the first hexadecimal code representing 3 bits only) and 00 3B EE 11 68 00 00.

An evolutionary algorithm is used for this task. Among the discovered automata some also accept glider guns. One of these automata, called R_0, is selected and attempts are made to demonstrate its universality. Unfortunately, no eater can be found in R_0. A second search using an evolutionary engine is then made among all automata accepting the glider gun that appeared in R_0. An eater is found in an automaton called R.

Then, several steps are used to prove the universality of R. The first step is to simulate a NAND gate as well as a means to intersect and synchronize glider streams, in order to be able to interconnect several gates into a logic circuit.

Then, a cell of Life is simulated using the tools elaborated above. Finally, it must be shown that it is possible to tile a 2D space with simulated interconnected Life cells to be able to simulate Life in R.

As Life was demonstrated to be computationally universal in the Turing sense [9], R is therefore also computationally universal in the Turing sense.

4 FINDING AN AUTOMATON ACCEPTING GLIDERS

The search space is the set $\mathcal{I}$ described in Section 2. An automaton of this space can be described by telling what will become of a cell in the next generation, depending on its neighbours. An individual is an automaton coded as a bitstring of 102 booleans (cf. Figure 2) representing the values of a cell at the next generation for each neighbourhood.

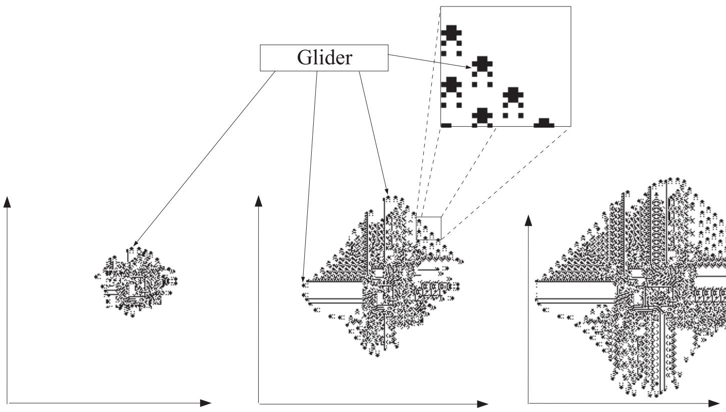


FIGURE 3
Result of the evolution of a random configuration of cells by the rule 00 80 80 00 00 00 00, 02 FF FD FF FF FF FF at generation 10, 20, and 30.

4.1 Using an evolutionary algorithm to find an automaton accepting gliders

Fitness Function Several fitness functions had to be tried before a correct result was found. One of the first fitness functions tried to maximise the number of gliders to appear during the evolution of a random configuration of cells by the tested automaton. Unfortunately, with this fitness function, many evolved automata not only accepted gliders, but also evolved a random configuration of cells into a constantly growing population that would go infinite (cf. Figure 3 for a typical evolution of such rules).

After several other attempts, the best fitness function simply attempts to maximize the number of gliders $\times$ the number of periodic patterns that appear during the evolution of a random configuration of cells by the tested automaton. (A more detailed description of the gliders and periodic patterns detector (inspired by Bays [19]) can be found in [20].)

Initialisation The 102 bits of each individual are initialized at random.

Genetic Operators The mutation function simply undertakes mutating one bit among 102, while the recombination is a single point crossover with a locus situated exactly on the middle of the genotype. This locus was chosen since the first 51 neighbourhoods determine the birth of cells, while the other 51 determine how they survive or die.

Evolution Engine It is very close to a $(\mu + \lambda)$ Evolution Strategy, although on a bitstring individual, and therefore without adaptive mutation : the population is made of 20 parents that are selected randomly to create 20 children by mutation only and 10 others by recombination. As in a straight

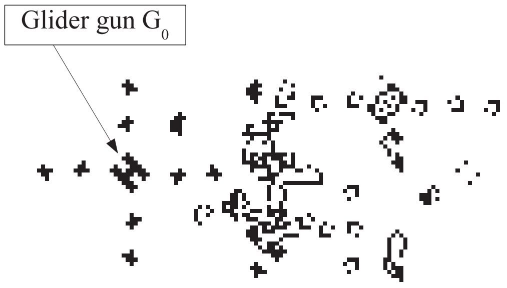


FIGURE 4
An evolution of a random configuration of cell by R_0 showing G_0.

ES+, the 20 best individuals among the 20 parents + 30 children are selected to create the next generation.

Stopping Criterion The algorithm stops after 200 generations, which, on a 3.2 GHz PC takes around 5 minutes to complete.

4.2 The R_0 automaton: an experimental result

The algorithm described above provided several automata accepting gliders. Particular discovered gliders are described in Appendix 1.

Among the discovered automata, some would surprisingly generate glider guns for nearly every evolution of a random cell configuration. The first of theses automata, R_0, is picked up as a potential candidate for a universal automaton. Its 102 different neighbourhoods can be visually presented as in Figure 2. Figure 4 shows the glider gun G_0 that keeps appearing spontaneously from virtually any random configuration of cells.

5 LOOKING FOR AN "EATER"

The automaton R_0 that was discovered in the previous section accepts gliders and a glider gun G_0. The gun G_0 fires one glider towards each cardinal point whereas the gun of Life fires just one glider. In order to use the glider gun G_0 in the same way as the glider of Life, three glider streams must be cancelled. An eater could be used to suppressed the undesirable streams, (cf. Figure 5) meaning that an eater is now discovered.

Ideal eaters are periodic patterns that, after the absorption of a glider, can resume their original shape and position quickly enough to absorb another arriving glider (cf. Figure 5).

5.1 Second evolutionary algorithm

As unfortunately, no eater could be found in R_0, it was decided to keep the G_0 glider, and explore all automata that would accept glider G_0 in order to

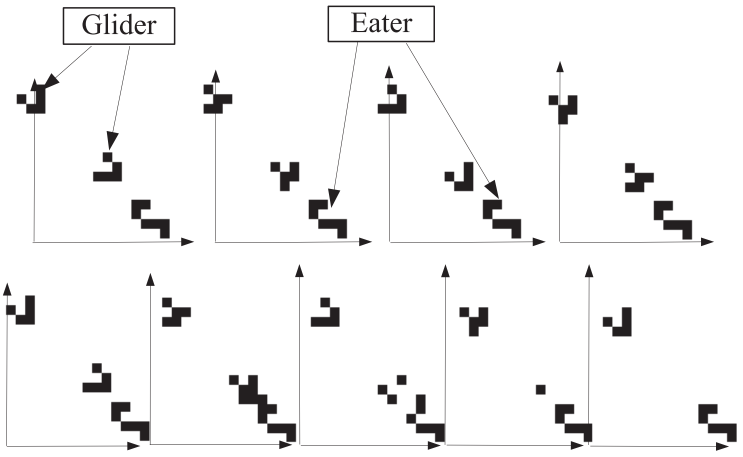


FIGURE 5
Eater in the Game of Life.

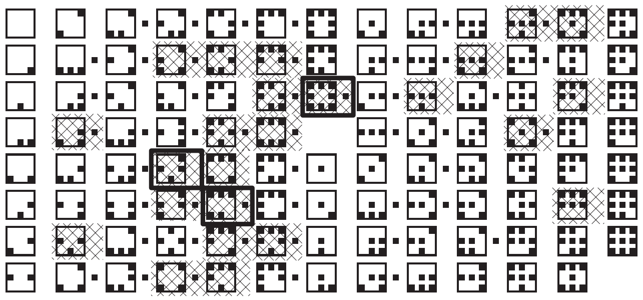


FIGURE 6
A black cell on the right of the neighbourhood indicates a future central cell. The R automaton can then be represented by 00 03 8E FF BC B7 F9, 00 3B EE 11 68 00 00. The neighbourhoods which are in a box are the ones for which the corresponding value is different for R_0 and the neighbourhoods in the hatching area is not used by G_0.

find an automaton that would accept both G_0 and an eater (cf. [12]). The resulting search space was determined by deterministically finding which of the 102 neighbourhoods of automaton R_0 were needed for gun G_0 to operate normally. It turns out that G_0 uses 81 different neigbourhoods, meaning that the output of only 21 other neighbourhoods (102-81) could be changed to get an automaton accepting an eater (cf. Figure 6). The search space for this second evolutionary algorithm contains “only” 2^{21} elements.

An eater being a periodic pattern, a collection of 10 small periodic patterns of R_0 appearing frequently *and using only* neighbourhoods among the 81 ones necessary for G_0 were chosen (cf. Figure 7). Since theses

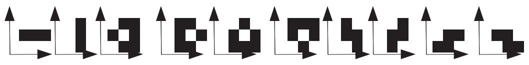


FIGURE 7
10 small periodic patterns of R_0 appearing frequently, using the same neighbourhoods as the G_0 glider.

periodic patterns were only using the unchanging 81 neighbourhoods, they were sure to appear in all of the remaining 2^{21} automata also implementing G_0.

Finally, to determine if a periodic pattern is an eater, one needs a fitness function that takes the form of a kind of "crash test:" each periodic pattern is positioned in front of a stream of gliders, and its fitness is simply the number of crashes it survives.

The number of possibilities being quite large (10 patterns to be tested in different relative positions with reference to a stream of gliders among 2^{21} different automata), a second evolutionary algorithm was therefore created, with the following characteristics:

Individual Structure An individual is made of:

- a bitstring of 21 elements determining one automaton among 2^{21},
- an integer between 1 and 10, describing one pattern among the 10 chosen periodic patterns,
- the relative position of the pattern relatively to the stream of gliders, coded by two integers, x and y, varying between $[-8, 8]$ and $[0, 1]$.

Individuals initialisation The bitstring part is initialised at random, the pattern ID is a random integer between 1 and 10, and the position of the eater wrt the glider stream is initialised at within the correct interval for x and y.

Fitness Function Number of gliders stopped by an individual.

Genetic Operators The only operator is a mutator, since no really "intelligent" recombination function could be coined. The mutator is therefore called on all created offsprings and can either:

- mutate one bit in the bitstring,
- choose any pattern among the 10 available,
- move the position of the pattern by ± 1 within the defined boundaries for x and y.

Evolution Engine It is this time closer to an Evolutionary Programming engine, since it has no crossover, although the EP tournament was not implemented. 25 children are created by mutation of 25 parents. Among the 50 resulting individuals, the 25 best are selected to create the next generation.

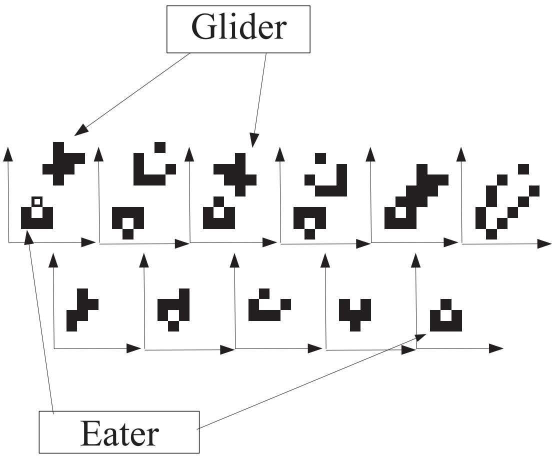


FIGURE 8
Eater of R in action, found by a second evolutionary algorithm.

Stopping Criterion Discovery of an eater that would survive 50 000 collisions.

5.2 Discovering an eater and the *R* automaton: an experimental result

This second evolutionary algorithm allowed to discover the automaton R, accepting both the glider gun G_0 of R_0 and the eater shown Figure 8.

Figure 6 shows that the automaton R is different from R_0 by just by three bits (FC 97 F8 $\rightarrow$ BC B7 F9) :

1**1**111100 10**0**10111 1111100**0**
1**0**111100 10**1**10111 1111100**1**

The role of the modified bits between R and R_0 can be observed in Figure 9 which shows the discovered R eater, evolving by the R_0 rule (in which it is not an eater). In this figure, cell A (shown on the right of the figure) comes to life with the R_0 rule, while it stays dead with R.

Three eaters are used to suppress the three extra streams of gliders produced by the glider gun G_0 in order to build the gun G (cf. Figure 10).

6 UNIVERSALITY OF THE *R* AUTOMATON

The R automaton now accepts an eater and a single-stream glider gun *à la* Life. We now need to prove the universality of R:

1. The first step is to find a configuration of glider guns and eaters that can simulate a NAND Gate, both to prove logic universality, but also because our simulation of a cell of Life needs a NAND gate.

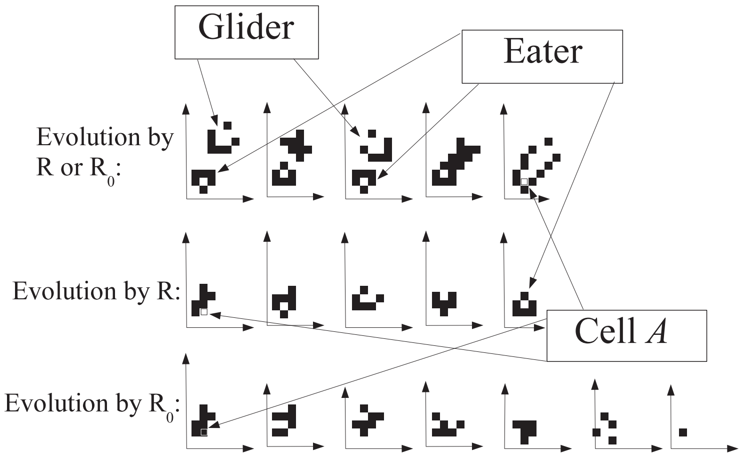


FIGURE 9
The first line shows the eater of R evolving by the rules R_0 or R. Nothing changes until generation 6 where cell A is white by R, and black by R_0. The eater returns to its original state in R, while it is destroyed in R_0.

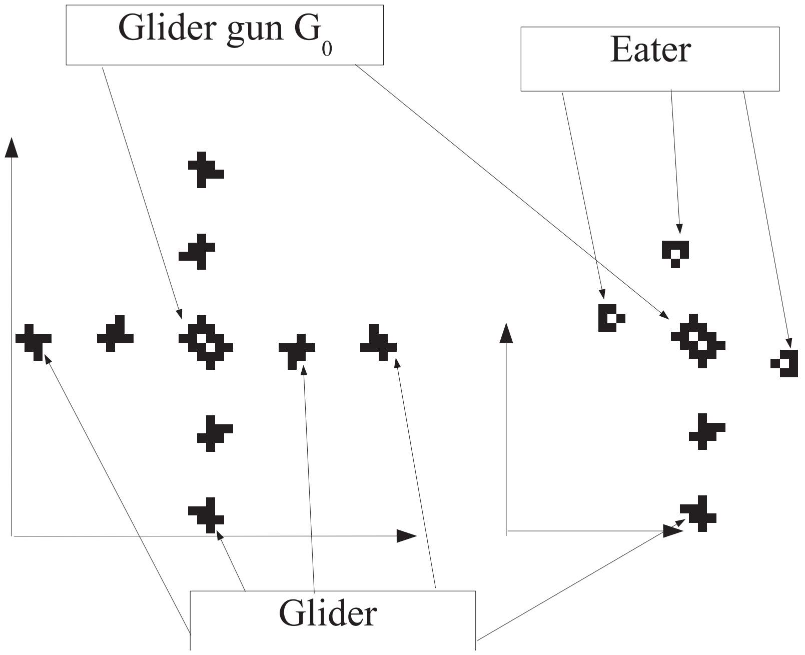


FIGURE 10
The glider gun G_0 emits four streams of gliders. In order to obtain a simple glider gun, three eaters are positioned around the G_0 gun to cancel three of the four streams, resulting in a G glider gun emitting one stream only.

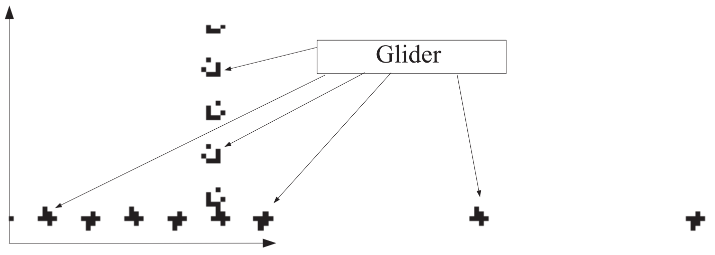


FIGURE 11
A 5-thinning collision of perpendicular streams in R.

2. The second step is to use the simulated NAND Gate to recreate one cell of Life.
3. The final step is to tile the 2D space of the Game of Life by identical simulation of identical cells in order to prove the universality of R.

6.1 Implementation of a NAND gate through stream collisions

Binary numbers can be represented as finite stream of gliders, where gliders represent 1s and missing gliders represent 0s.

The simulation of a NAND gate is based on collisions between streams of gliders. The different types of collisions are described below, before they are assembled to form a NAND gate.

Depending on the relative position of one stream with reference to the other, five different types of collisions are listed. The first four are collisions between perpendicular streams of gliders, while the last one describes a frontal collision.

Thinning Collision A thinning collision is a collision between two perpendicular streams of gliders which destroys one of the two streams, and lets a glider of the other stream survive every n gliders. A collision is n-thinning if one every n gliders survives the collision. Figure 11 shows a 5-thinning collision between two streams emitted by two guns. In the initial streams, gliders are spaced by 9 cells. Therefore, in the final remaining stream, gliders are spaced by 45 cells.

Selective-1 collision A selective-1 collision is a collision of two streams of perpendicular gliders which destroys one of the streams but lets the other one survive. After the collision, the gliders of the surviving stream are of another kind, called *large gliders* as shown in Figure 12. The front cells of the large gliders are identical to those of the standard gliders. Therefore, the speed and spacing of the large gliders is identical to the speed and spacing of the original standard gliders. Large gliders can be seen as being fat standard gliders.

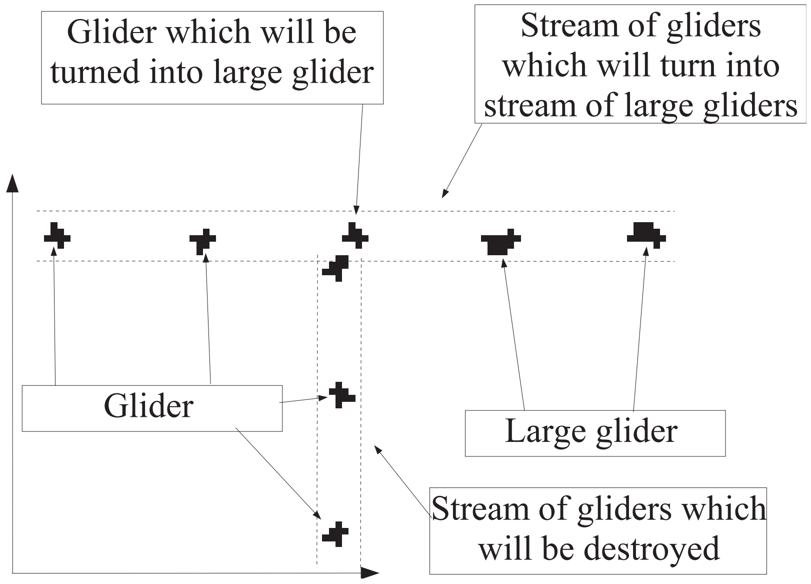

Glider which will be turned into large glider
Stream of gliders which will turn into stream of large gliders
Glider
Large glider
Stream of gliders which will be destroyed

FIGURE 12
Selective-1 collision between two gliders.

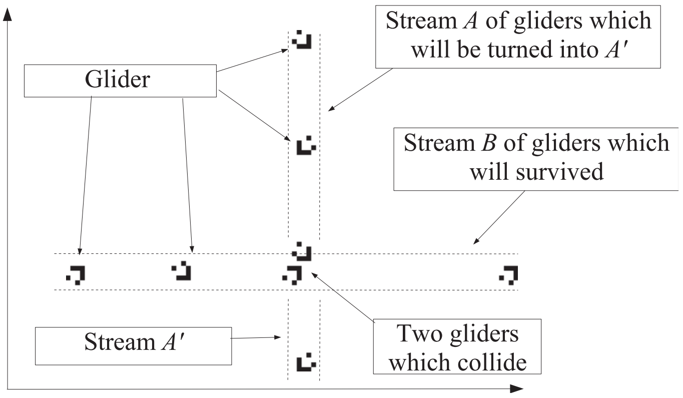

Stream *A* of gliders which will be turned into *A′*
Glider
Stream *B* of gliders which will survived
Stream *A′*
Two gliders which collide

FIGURE 13
Selective-2 collision between two gliders.

Selective-2 Collision A selective-2 collision is a collision of two streams of gliders which destroys one stream, called *A*, but lets the other one, called *B*, survive as shown in Figure 13. If there is a missing glider in stream *B* then the corresponding glider in stream *A* survives. So stream *B* is unchanged by this collision and the result is the following for stream *A*:

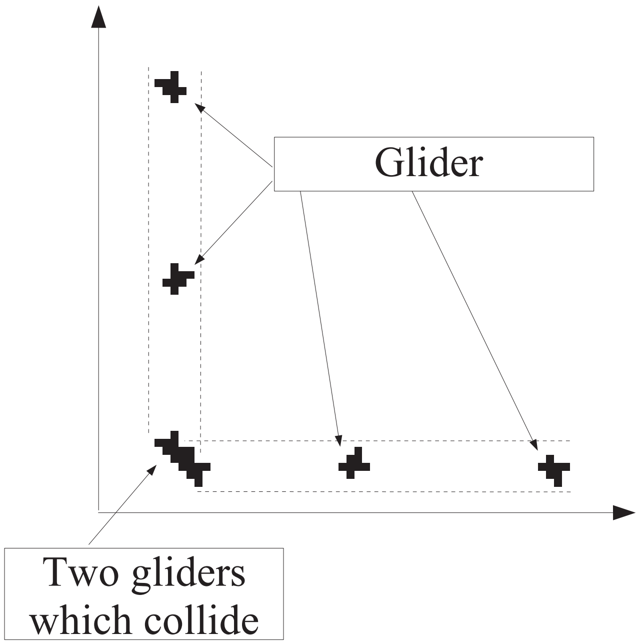


FIGURE 14
Annihilating collision between two gliders.

- If there is a missing glider in stream A, after the collision, there is a missing glider in the resulting stream A'.
- If there are gliders in both streams A and B, after the collision there is a missing glider in stream A'.
- If there is a glider in stream A and a missing glider in stream B, after the collision, there is a glider in stream A'.

So, the resulting stream A' become the result of the operation A *and* $\overline{B}$. If A is a full stream, then it becomes $\overline{B}$ whereas the stream B survives.

Annihilating Collision An annihilating collision is a collision of two perpendicular streams of gliders or large gliders that cancels both streams. If there is a missing glider or a missing large glider in one stream then the corresponding glider in the other stream survives. The result is the following for the two streams:

- If there is a glider in each stream, after the collision there is no glider in the resulting stream.
- If there is a missing glider in the one of the two streams, after the collision the corresponding glider survives in the other stream.

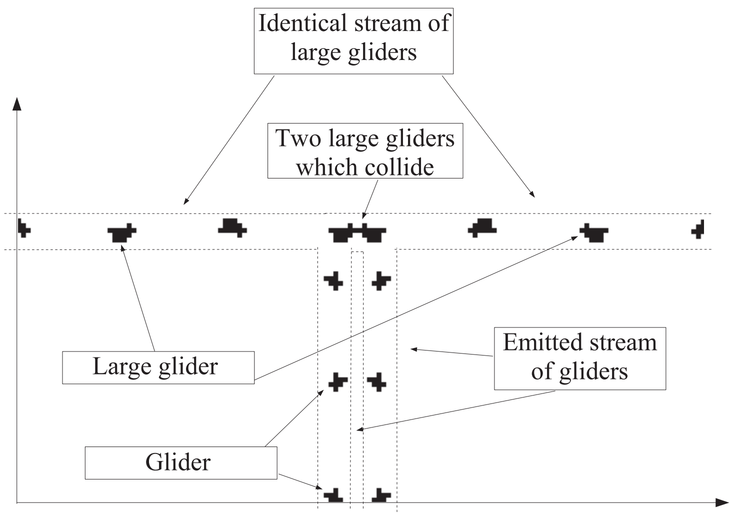

Identical stream of large gliders
Two large gliders which collide
Large glider
Emitted stream of gliders
Glider

FIGURE 15
Right angle collision between two large gliders every four generations.

- If there are a missing gliders in both streams, after the non-collision there are a missing gliders in the resulting streams (no gliders are created *ex-nihilo*).

Therefore, supposing that two streams A and B collide through an annihilating collision. The resulting stream A' is the result of the operation $A\ and\ \overline{B}$ and the resulting B' stream becomes $B\ and\ \overline{A}$. If stream A was a full stream, then A' equals $\overline{B}$ whereas the stream B is destroyed.

Right angle collision A right angle collision is a frontal collision of two identical streams of large gliders which produces two orthogonal streams of standard gliders going in the same direction, equal to the input streams as shown in Figure 15.

6.2 Concise notation for the description of cellular automata

The cellular automaton implementing a cell of Life would be too difficult to describe by showing groups of cells on a grid. Therefore, a much clearer analytical description was needed, that should also allow replicability of the contents of this paper.

In order to simplify the representation of a CA, one can think of it in terms of building blocks that can be represented by an analytical description, made of a letter (referring to a pattern) followed by three parameters (D, x, y), where D denotes a direction (North, East, South, West) and x, y are the coordinates of a specific cell of the pattern (marked in white cf. [21]). An arrow is added in graphic descriptions to help visualising the

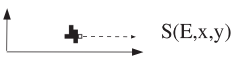


FIGURE 16
A glider stream and its analytical representation $S(E, x, y)$.

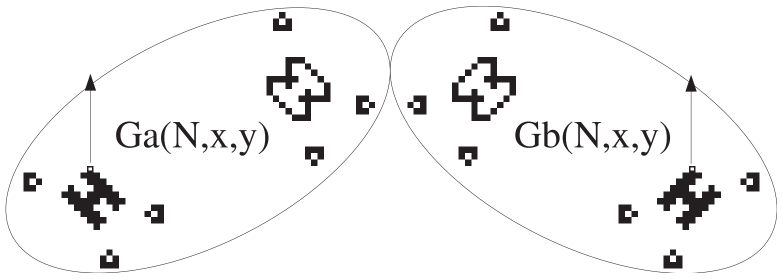


FIGURE 17
Two complex guns of R, viewed at generation 2. (x, y) are the coordinates of the white cell, whence an arrow is shooting.

pattern. The analytical description of a glider stream and a eater are given, in order to build a complex glider gun and a large glider gun:

Glider Stream $S(D, x, y)$**:** Figure 16 shows a glider stream $S(E, x, y)$, where x and y are the coordinates of the white cell whence an arrow is shooting.

Eater $E(D, x, y)$**:** The eater of Figure 8 can be identified as $E(N, x, y)$, where N denotes its northward orientation and x, y denote the position of the white cell.

Complex Glider Guns $Ga(D, x, y)$ **and** $Gb(D, x, y)$**:** The standard glider gun of Figure 4 shoots gliders spaced by only nine cells. Unfortunately the Selective-2 collision needs gliders that are more loosely spaced. In order to build a gun shooting such gliders, the thinning collision shown in Figure 11 is used. This results in a complex gun, shown in Figure 17, that shoots gliders spaced by 45 cells, which gives more slack to work on streams. Figure 17 shows two mirror instances of this gun ($Ga(S, x, y)$ and $Gb(S, x, y)$), used later on in this paper. The complex guns in other cardinal directions are obtained by rotation.

Large Glider Gun $L(D, x, y)$**:** Thanks to a selective-1 collision, a large gun shooting a large glider every 45 cells is made of two complex guns G shooting their stream perpendicularly (cf. Figure 18). The position of the resulting large gun is the position of the left Gb gun.

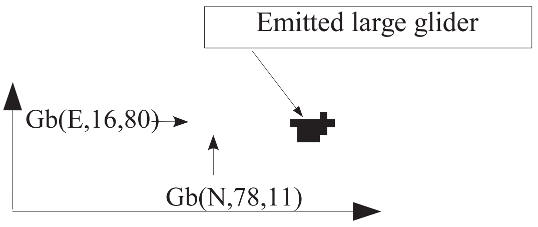


FIGURE 18
Schematics of a large glider gun $L(E, 16, 80)$, made of two complex Gb guns, with gliders emitted on the right.

6.3 Creation of a `NOT` gate

On Figure 19, a NAND gate is built thanks to the patterns described in the previous section. On this figure, the stream A ($S(S, 208, 197)$) is shown as a dotted line. A complex gun $Gb(E, 179, 200)$ creates a complementary duplicate stream $\overline{A}$ towards the East. The two outputs are redirected by complex guns $Gb(W, 253, 142)$, $Gb(S, 212, 65)$ and $Ga(S, 273, 262)$ until they are vertical again. Then, they are complemented into large gliders by the two guns of large gliders $L(E, 16, 80)$ and $L(W, 379, 80)$. When the frontal collision between the two streams of large gliders occurs, a complementary stream $\overline{A}$ is created towards the same direction as the original A stream while the other one is "eaten" by $E(N, 216, 65)$.

6.4 Simulation of one cell of Life

A single cell of Life can be implemented as a boolean function computing the value of a cell S at generation $n+1$ from its value and that of its eight neighbours $C_1 \ldots C_8$ at generation n.

The rules of the Life automaton are the following: a "living" cell dies at the next generation unless it has two or three neighbours. A dead cell comes alive at the next generation iff it has three neighbours in the current generation.

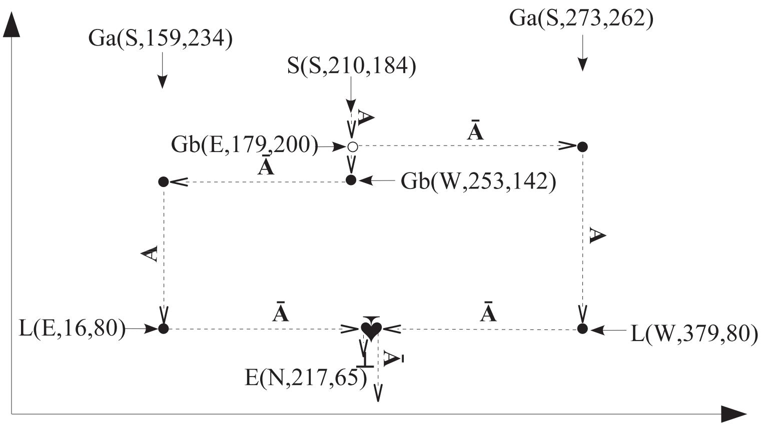
Supposing that the addition of $C_1 + \ldots + C_8$ gives a four bit number $n_3n_2n_1n_0$. Eight neighbours in state 1 give the number 1000 and the bit n_3 is equal to 1 just for this case.

A cell in state 0 or 1 surrounded by eight living neighbours always dies at the next generation, which is also the case if a cell is surrounded by eight dead neighbours.

This means that both cases (eight living neighbours or eight dead neighbours) will yield the same result for the central cell: 0.

Therefore, the n_3 bit can be ignored, since, if its value is 1, the value of n_2, n_1 and n_0 is always 0, which is also the case if the cell has 0 living neighbours.

Thanks to this small simplification, the rules of Life can be simply expressed by the formula $S_{n+1} = \overline{n_2} \,.\, n_1 \,.\, (S_n + n_0)$, which can be translated



○ : Selective-2 collision
● : Annihilating collision
♥ : Right angle collision
⊥ : Collision between a glider and an eater

FIGURE 19
Complementation of stream A: the analytical representation for the NOT gate is $\{L(E, 16, 80), Gb(S, 212, 65), Gb(W, 241, 36), Gb(W, 253, 142), L(W, 379, 80), Ga(S, 159, 234), Gb(E, 179, 200), Ga(S, 273, 262), S(S, 1, 210, 184), E(N, 219, 78)\}$.

into a combination of NAND gates. This function, implementing the value of a cell of Life at generation $n+1$ is implemented in Figure 20, and analytically described using R in appendix 2.

6.5 Simulation of Life

Now that we have a simulation of a single cell of Life, it is necessary to tile a surface with such identical cells, each interconnected with its 8 neighbours in a way that they remain functional.

In order to connect a cell to its 8 neighbours, it is necessary to redirect output streams, so that they can become the input streams of a neighbouring cell. This means that it should be possible to intersect and synchronize glider streams.

6.5.1 Intersection of streams

Thanks to complex guns, gliders in a stream can be separated by 45 cells. This means that it is possible to have two streams cross each other without any interference. If, for a synchronisation reason, interference cannot be avoided, [22] shows how a stream intersection can be realised without interference by an special combination of NAND gates as shown in the Figure 21.

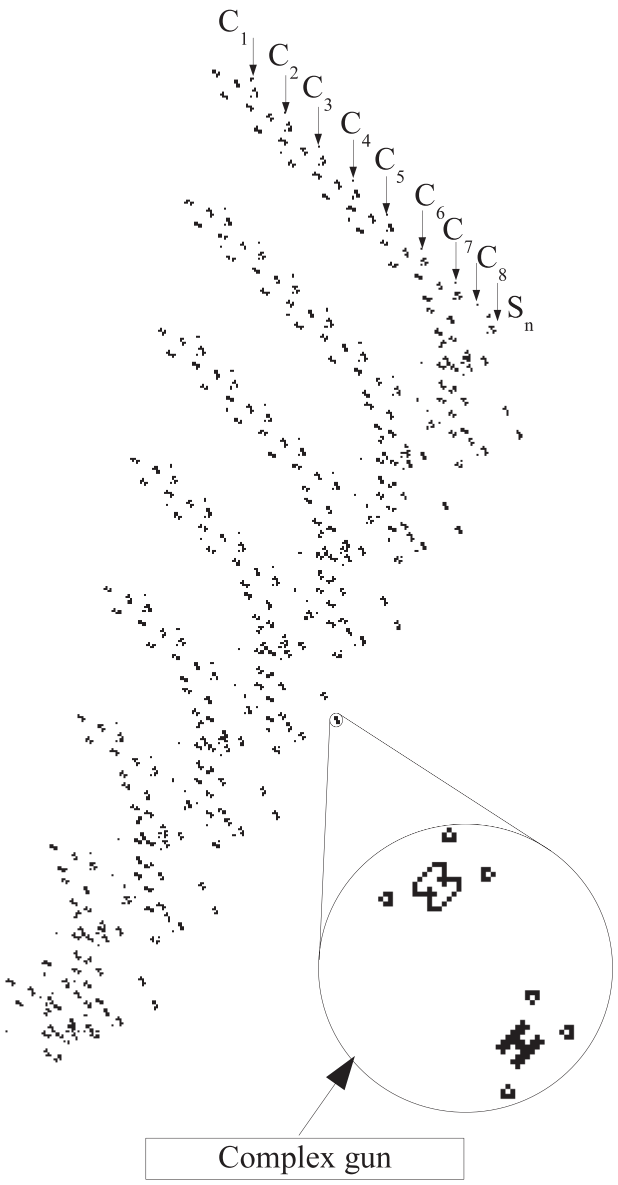


FIGURE 20
Simulation of a cell of Life by R. The analytical description is given in appendix 2.

6.5.2 Synchronisation

It is important to be able to delay one stream w.r.t another, in order to synchronise them properly just before they enter a logic gate, for instance.

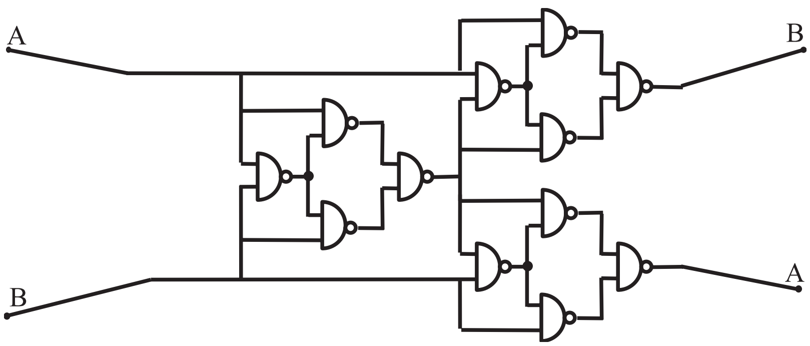


FIGURE 21
Cunning combination of Nand gates that allows to intersect two streams without interference.

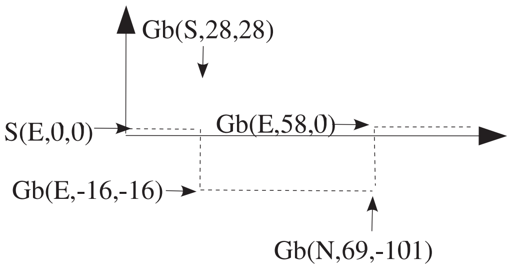


FIGURE 22
Stream temporisation for synchronisation purposes.

This can be done precisely by diverting four times the stream to be delayed with orthogonal guns (cf. Figure 22).

6.5.3 Simulation of Life in R

A single cell of Life can be implemented as a boolean function computing the value of a cell S at generation $n+1$ from the value of its eight neighbours $C_1 \dots C_8$ at generation n.

All cells being identical, the inputs of a cell must physically correspond to the outputs of its neighbours and conversely. If this is the case, the way a cell receives the state of its neighbours can be induced from the way it sends its own state to its neighbours, which is what is described below and in Figure 23.

It is straightforward for a cell to send its state to its cardinal neighbours C_2, C_4, C_5, C_7. Sending its state to neighbours C_1, C_3, C_6, C_8 is however more tricky, since those neighbours are situated diagonally. This is done by passing the information to their neighbours C_2 and C_7. Therefore, one can see in Figure 23 that the state S of the cell is sent three times to C_2 and three times to C_7, so that C_2 (resp. C_7) can keep one stream for its

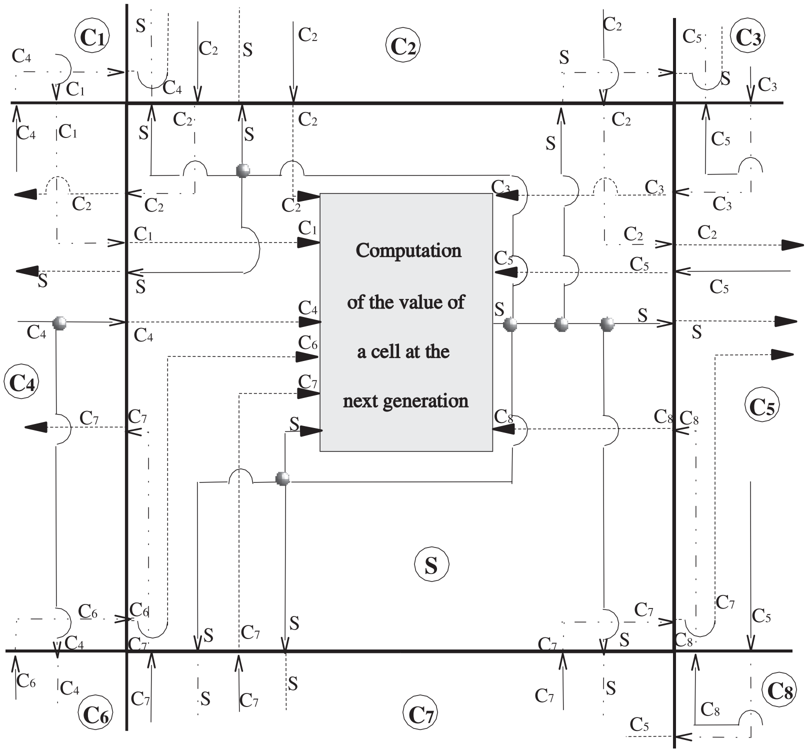


FIGURE 23
Diagram of the interconnection of a cell of Life implemented in R.

own use, and pass the two others to its horizontal neighbours, C_1 and C_3 (resp. C_6 and C_8).

S being itself a top neighbour of cell C_7, one sees how the state of C_7 is passed over to C_4 and C_5 in the same way that C_2 will pass over the information of the state of S to C_1 and C_3.

7 SYNTHESIS AND PERSPECTIVES

An extensive bibliographic research seems to show that the R automaton is the first 2D 2 state dynamical universal (in the Turing sense) automaton, other than Life, discovered in $\mathcal{I}$.

Further goals are now to give a more complete answer to Wolfram's problem "How common are computational universality and undecidability in cellular automata?" by finding whether other universal automata than Life and R exist, and how common they are. Then, another domain that seems worth exploring is how this approach could be extended to automata with more than 2 states.

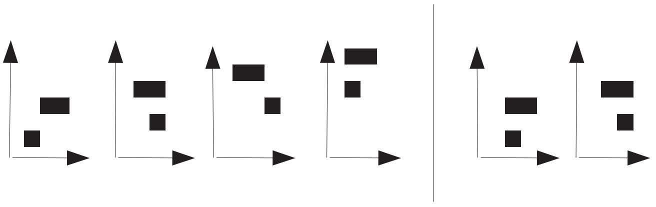


FIGURE 24
Gliders which appear the most often.

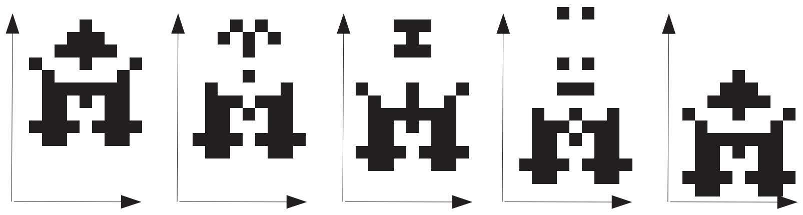


FIGURE 25
Largest discovered glider.

The proof of universality of R can be generalized to other automata accepting glider guns, meaning that other automata accepting glider guns may also be potential candidates for being universal automata.

This provides an element of answer to the question of the frequency of apparition of universal automata related to emergence of computation in complex systems with simple local behaviour[7].

Finally, the study of the construction of an automatic system of selection/ discovery of such type of automata based on evolutionary algorithms is far more interesting. The discovery of new universal cellular automata can lead to a new classification of cellular automata and give an element of the answer to Wolfram's problem: "What overall classification of cellular automaton behaviour can be given?"

APPENDIX 1

The evolutionary algorithm described in section 4 provided several automata accepting gliders. Particular discovered gliders accepting by at least one of the found automata are described in this annexe. The discovered gliders which appear the most often have three cells (cf. Figure 24). This is the minimum, because two cells define a line but not a direction. It agrees with the idea that the simplest patterns appear more easily).

Figure 25 shows the largest discovered glider. This glider appeared only once.

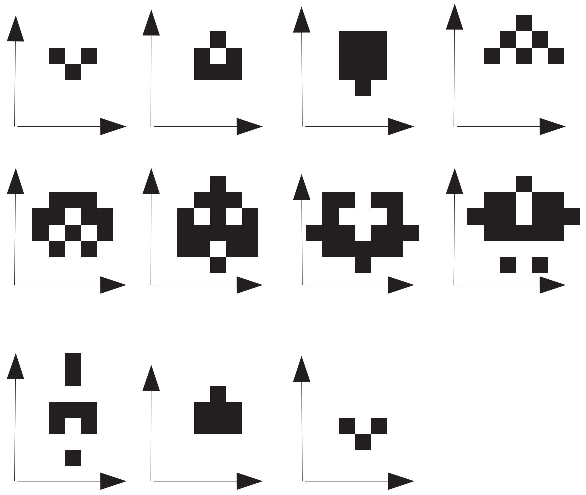


FIGURE 26
Slowest discovered glider (moves one cell in 10 generations).

The most common period of the discovered gliders is two. The largest period for the discovered gliders is 10 (cf. Figure 26). This glider moves forward of one cell so it is also the slowest discovered glider.

A glider moves x steps along the X axis and y steps along the Y axis in one period. For the overwhelming majority of gliders, x and y follow one of these two rules:

- $x = y$
- $x = 0$ *or* $y = 0$

However, Figure 27 shows a glider which does not follow any of these rules. It mean that this glider does not move horizontally or diagonally.

Some glider guns have been found in discovered automata other than R. Figure 28 shows a glider gun with a period of 19 that shoots a glider to the left at generation 10, and another glider to the right at generation 20.

APPENDIX 2

This appendix contains an analytical description of a cell of the game of life in the R automaton, for replicability. Let us define X, Y and Z as:

$X(S, 212, 65) = \{Gb(E, 18, 91), Gb(N, 80, 25), Gb(S, 212, 65), Gb(W, 241, 36), Gb\ (W, 253, 142), Gb(N, 370, 23), Gb(W, 433, 93), Ga(S, 159, 234), Gb(E, 179, 200),\ Ga(S, 273, 262), S(S, l, 210, 184), E(E, 292, 159), E(S, 219, 78)\}$,

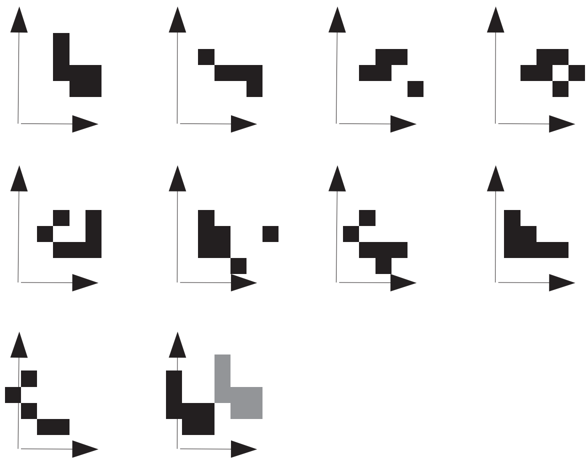


FIGURE 27
Glider that is moving neither horizontally nor diagonally. The original position of the glider is shown in grey on the last generation.

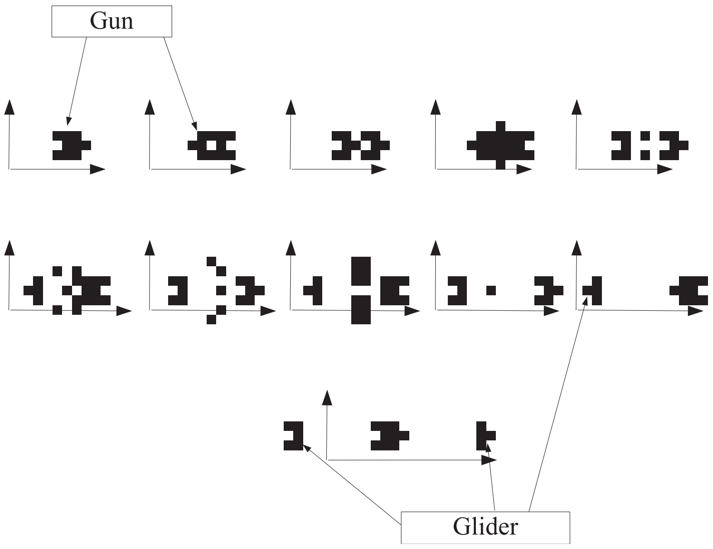


FIGURE 28
A glider gun at generations from 1 to 10 at the top and 19 at the bottom.

$Y(E, 1, 1) = \{Gb(E, 1, 1), Ga(S, 99, 189), Ga(S, 173, 520), Gb(W, 184, 104), E(E, 183, 484)\}$, $Z(E, 1, 1) = \{Y(E, 1, 1), X(S, 425, 155), Gb(E, 721, 91), Gb(W, 843, 213), Ga(S, 749, 299), Ga(S, 839, 630), Ga(S, 1019, 749), Gb(W, 1246, 261), E(E, 1229, 276), Gb(E, 990, 180), Gb(E, 965,$

277), $Ga(S, 1071, 350)$, $Ga(S, 1145, 457)$, $E(E, 1161, 394)$, $Ga(S, 423, 455)$, $E(S, 427, -47)\}$.

A cell of the game of life can be described in R as:
LifeCell(E, 1259, −436) = $\{Z(E, 1, 1)$, $Z(E, -759, 486)$, $Z(E, -1519, 971)$, $Z(E, -2279, 1456)$, $Z(E, -3039, 1941)$, $Z(E, -3799, 2426)$, $Z(E, -4559, 2911)$, $Y(E, -269, -89)$, $Y(E, -1028, 397)$, $Y(E, -1298, 127)$, $Y(E, -1787, 883)$, $Y(E, -2057, 613)$, $Y(E, -2327, 343)$, $Y(E, -2546, 1369)$, $Y(E, -2816, 1099)$, $Y(E, -3086, 829)$, $Y(E, -3356, 559)$, $Y(E, -3305, 1855)$, $Y(E, -3575, 1585)$, $Y(E, -3845, 1315)$, $Y(E, -4115, 1045)$, $Y(E, -4385, 775)$, $Y(E, -4064, 2341)$, $Y(E, -4334, 2071)$, $Y(E, -4604, 1801)$, $Y(E, -4874, 1531)$, $Y(E, -5144, 1261)$, $Y(E, -5414, 991)$, $Y(E, -4823, 2827)$, $Y(E, -5093, 2557)$, $Y(E, -5363, 2287)$, $Y(E, -5633, 2017)$, $Y(E, -5903, 1747)$, $Y(E, -6173, 1477)$, $Y(E, -6443, 1207)$, $Ga(S, 843, 33)$, $Ga(S, 933, 123)$, $Ga(S, 1203, 213)$, $X(S, 1205, -84)$, $Gb(E, 663, -147)$, $Y(E, 719, -437)\}$.

The input streams must arrive at coordinates –6227,988 for Sn, –5957,1258 for $C8$, –5687,1528 for $C7$, –5417,1798 for $C6$, –5147,2068 for $C5$, –4877,2338 for $C4$, –4607,2608 for $C3$, –4427,2788 for $C2$, –4337,2878 for $C1$. The stream for $Sn + 1$ comes out at coordinates 1259,–436 after about 21600 generations.

REFERENCES

[1] Wolfram, S. (1984). Universality and complexity in cellular automata. In *Physica D 10*, 1–35.

[2] Gardner, M. (1970). The fantastic combinasions of john conway's new solitaire game "life". *In Scientific American.*

[3] Gardner, M. (1971). On cellular automata, self-reproduction, the garden of eden, and the game of life. In *Scientific American 224*, 112–118.

[4] Dytham, C., Shorrocks, B. (1992). Selection, patches and genetic variation: A cellular automata modeling drosophila populations. In *Evolutionary Ecology 6*, 342–351.

[5] Epstein, I.R. (1991). Spiral waves in chemistry and biology. *In Science 252.*

[6] Lotti, Ermentrout, G. and Margara, L. (1993). Cellular automata approaches to biological modeling. In *Journal of Theoretical Biology 60*, 97–133.

[7] Langton, C.G. (1996). *Artificial life: An overview*. Cambridge, Massachusetts: MIT Press.

[8] Wolfram, S. (1985). Twenty problems in the theory of cellular automata. In *Physica Scripta* 170–183.

[9] Conway, E., Berlekamp, J. and Guy, R. (1982). Winning ways for your mathematical plays. *Academic press*, New York.

[10] Packard, S. and Wolfram. N. (1985) Two-dimensional cellular automata. In *Journal of Statistical Physics 38*, 901–946.

[11] Rendell, P. (2002). Turing universality of the game of life. Andrew Adamatzky (ed.), *Collision-Based Computing*, Springer Verlag.

[12] Banks, E.R. (1971). Information and transmission in cellular automata. PhD thesis, MIT.

[13] Margolus, N. (1984). Physics-like models of computation. In *Physica D 10*, 81–95.

[14] Lindgren, K. and Nordahl, M. (1990). Universal computation in simple one dimensional cellular automata. In *Complex Systems 4*, 299–318.

[15] Ogiro, K., Morita, Y., Tojima, I. and Katsunobo.T. (2002). Universal computing in reversible and number-conserving two-dimensional cellular spaces. Andrew Adamatzky (ed.), *Collision-Based Computing*, Springer Verlag.

[16] Adamatzky, A. (1998). Universal dymical computation in multi-dimensional excitable lattices. In *International Journal of Theoretical Physics 37*, 3069–3108.

[17] Goldberg, D.A. (1989). Genetic algorithms. *Search, Optimization, and Machine Learning*. Addison-Wesley.

[18] Heudin, J. (1996). A new candidate rule for the game of two-dimensional life. *Complex systems 10*, 367–381.

[19] Bays, C. (1987). Candidates for the game of life in three dimensions. In *Complex Systems 1*, 373–400.

[20] Bailleux, E., Sapin, O. and Chabrier, J. (2004). Research of complex forms in the cellular automata by evolutionary algorithms. In *EA03. Lecture Notes in Computer Science 2936*, 373–400.

[21] Bailleux, E., Sapin, O. and Chabrier, J. (2003). Research of a cellular automaton simulating logic gates by evolutionary algorithms. In *EuroGP03. Lecture Notes in Computer Science 2610*, 414–423.

[22] Dewdney, A. (1984). *The planiverse*. Poseidon Press.