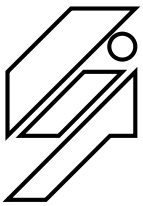


Laboratoire de l'Informatique du Parallélisme

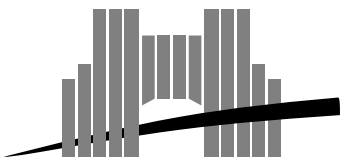
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

The Game of Life: universality revisited

B. Durand, Zs. Róka

January 1998

Research Report N^{O} 98-01



Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : (+33) (0)4.72.72.80.00 Télécopieur : (+33) (0)4.72.72.80.80
Adresse électronique : lip@lip.ens−lyon.fr

The Game of Life: universality revisited

B. Durand, Zs. Róka

January 1998

Abstract

The Game of Life was created by J.H. Conway. One of the main features of this game is its *universality*. We prove in this paper this universality with respect to several computational models: boolean circuits, Turing machines, and two-dimensional cellular automata. These different points of view on Life's universality are chosen in order to clarify the situation and to simplify the original proof. We also present precise definitions of these 3 universality properties and explain the relations between them.

Keywords: Game of Life, cellular automata, universality

Résumé

Le jeu de la Vie a été inventé par J.H. Conway. Un des principaux interêts de ce jeu est son universalité. Nous la prouvons ici dans différents models: les circuits booléens, les machines de Turing, les automates cellulaires de dimension 2. Ces différents points de vue sont pris pour clarifier la situation et simplifier les preuves originales. Nous donnons aussi des définitions précises de ces 3 propriétés d'universalité et expliquons leurs différences.

Mots-clés: Jeu de la Vie, automates cellulaires, universalité

The Game of Life: universality revisited

Bruno Durand,
LIP, ENS Lyon,
46 allée d'Italie, 69364 Lyon Cedex 07, France,
e-mail: Bruno.Durand@ens-lyon.fr

Zsuzsanna Róka
LRIM - IUT de METZ
Département Informatique
Ile du Saulcy
57045 Metz cedex, France
e-mail: roka@iut.univ-metz.fr

January 13, 1998

Abstract

The Game of Life was created by J.H. Conway. One of the main features of this game is its *universality*. We prove in this paper this universality with respect to several computational models: boolean circuits, Turing machines, and two-dimensional cellular automata. These different points of view on Life's universality are chosen in order to clarify the situation and to simplify the original proof. We also present precise definitions of these 3 universality properties and explain the relations between them.

Table of Contents

1 Introduction

The *Game of Life* (Life for short) is a cellular automaton introduced by J.H. Conway in 1970: it can be seen as a game played on an infinite grid. At any time some of the cells are alive and the others dead. Which cells are live at time 0 is up to the player who chooses the initial configuration. Then the player observes the evolution of the configuration: the state of each cell at any time follows inexorably the previous one according to the rules of the game (see Section 2). Conway proved in [2] the computational universality (let us say Turing-universality to avoid confusion in the remainder) of the game by defining some special patterns able to simulate the components of a real computer. Independently and simultaneously, it seems that Gosper obtained the same result but we could not have access to this — unpublished as far as we know — proof.

We focus on three aspects of Conway's proof of Turing-universality. First, one point is very complicated and difficult to check: the construction of a memory stack. Second, as far as we could understand it, it contains an error discussed in Section 3.3. Third, the author did not explain precisely what he means by "universality". It could be at first sight considered as a minor problem since it is also the case for instance in the works of Banks [1], Nourai and Kashef [11] and Smith [7]. As a matter of fact, all authors we could read on this topic did not defined what they meant by Turing-universality. This point may seem of low interest since the definition is usually straightforward, but in the case of simulation by cellular automata, it is a *very delicate* point because cellular automata transform infinite objects — versus finite in standard models. One can find in literature *false* results because of definition problems. Such a definition can even be extremely difficult to establish when combined with self-reproduction which is also hard to define [16, 3]. Thus, we give first a precise definition of this notion.

In this paper, we also prove a stronger universality property: we prove that life is *intrinsically* universal, *i.e.* that it can simulate any 2-dimensional cellular automaton. There is also a problem of definition for this notion of simulation, even if it is broadly used in literature. We base our work on the definition of simulation presented in [14]. The key point is that this definition is compatible with Turing-universality: if a Turing-universal cellular automaton is simulated by another cellular automaton, the latter is also Turing-universal. This intrinsic universality is implicitly used by Banks in [1] but it seems that he did not get the relationship with Turing-universality — which he also proves in a different way. This author focuses on the problem of the difference between finite and infinite initial configurations, which is indeed the key point of the definition of Turing-universality as explained in Section 4. The intrinsic universality allows to simplify proofs and, in our case, to prove directly the Turing-universality of Life without the difficult construction of the memory stack. This approach is also used in [5].

More precisely, the universality is proved in 3 different contexts: first for boolean circuits, then we prove it for Minsky machines (it corresponds to Turing-universality), and *in fine* we prove this universality for 2D cellular automata(intrinsic universality).

The paper is organized as follows: first in Section 2, we recall the rules of the game and show some basic configurations that are used as basic construction tools. In Section 3 we prove that any boolean circuit can be simulated by these basic patterns (our first kind of universality). In Section 4 we give and discuss a precise definition of Turing-universality for cellular automata. We give a hint of Conway's proof without entering the technical details of the construction of the memory stack. Section 5 is devoted to intrinsic universality: we propose adequate definitions and prove that Life is intrinsically universal.

2 A basic survey on Life

In this section we present a few important patterns and some reactions between them without modifying the book [2]. Then, in Section 5, we shall use them for constructing some special configurations to prove universality.

Life is a game played on an infinite grid. At any time, some cells are live and the others are dead. Each cell is surrounded by eight neighbors, forming the so-called Moore neighborhood (see Figure 1). At every time step, each cell can change its state, in a parallel and synchronous way according to the following local rules:

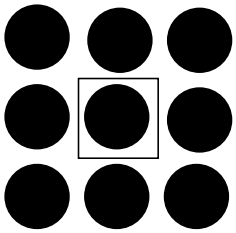


Figure 1: The Moore neighborhood

Birth: a cell that is dead at time t becomes live at time $t+1$ if and only if three of its eight neighbors were live at time t (for an example, see Figure 2a).

Survival: a cell that was live at time t will remain live if and only if it had just 2 or 3 live neighbors at time t (see Figure 2b).

Death

by overcrowding: a cell that is live at time t and has 4 or more of its eight neighbors live at t will be dead by time $t+1$ (see Figure 2c).

by exposure: a cell that has only one live neighbor, or none at all, at time t, will be dead at time $t+1$ (see Figure 2d).

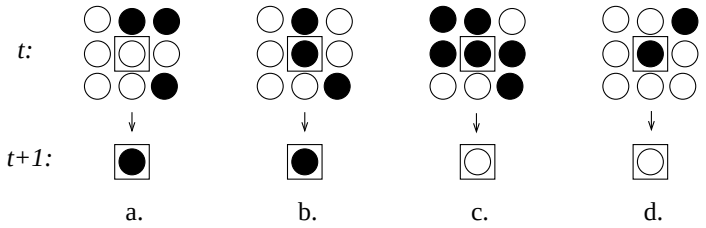


Figure 2: Examples for rules

Alternative definition

Let us now consider that the center cell belongs to the set of neighbors (*i.e.* that the neighborhood of a cell consists of nine cells). Let us denote by n the number of live cells in this neighborhood. Then the rules of the game are:

- $n \leq 2$: the cell dies,
- $n = 3$: the cell lives,
- $n = 4$: the cell stays in the same state,
- $n \geq 5$: the cell dies.

We can remark that Life can be considered as a two-dimensional cellular automaton over the alphabet $\{0, 1\}$.

Definition 1 *We call* pattern *a configuration that contains only a finite number of live cells.*

There can be many different types of evolutions for a given pattern. The pattern can turn into a stable configuration (which does not change) after a finite number of time steps, or its evolution can be dynamical. Some stable patterns can be seen in Figure 3. In the patterns of the present paper, we feature live cells by black bullets.

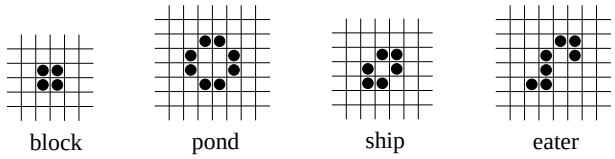


Figure 3: Stable patterns

Among patterns having dynamical evolution, we distinguish two cases:

- a pattern can have a periodical behavior: it repeats itself with period greater than one. Then, either its whole evolution needs a bounded room on the board (see an example in Figure 4), or it travels across the plane like the glider shown in Figure 5.

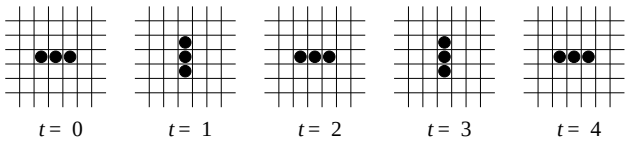


Figure 4: The blinker: a 2-period configuration

- a pattern can grow without limit as, for instance, Gosper's gun (see Figure 6) which emits a new glider every 30 generations (see Figure 7).

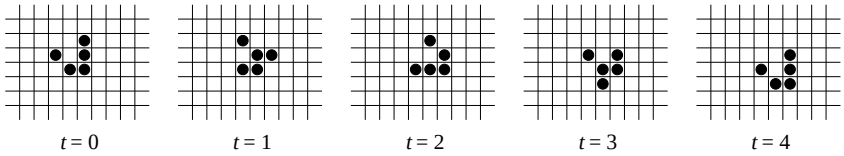


Figure 5: The glider: a 4-period South-East moving configuration

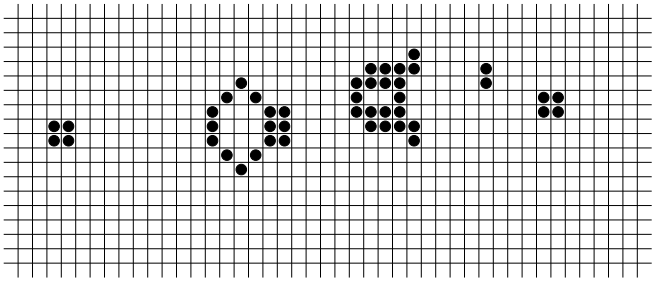


Figure 6: Gosper's gun

Other interesting patterns such as spaceships, guns etc can be found in literature. Here, we only present configurations used to prove the computational universality of the game.

Let us now study pattern evolution in general. A natural question arises:

Can one find an algorithm predicting the kind of evolution of a given pattern?

In [2], it is showed that even for the simple example of a straight line of k live cells, the evolution seems unpredictable. The answer to the more general question above is "no": the unpredictability of Life's evolutions is a consequence of its Turing-universality.

One can think about another natural question:

Given a configuration, does another configuration exist that precedes in time the first one?

The answer to this question is again "no": a configuration without predecessor is called a *Garden of Eden* configuration. There exist Garden of Eden configurations for life. One of them (shown in Figure 8) has been found by an M.I.T. group (R. Banks, M. Beeler, R. Schroeppel *et al.*)

To prove the existence of such a Garden of Eden configuration, we use the general theory of cellular automata:

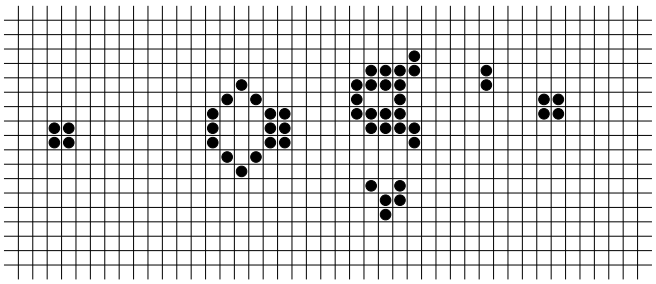


Figure 7: Gosper's gun: the first glider (after 30 iterations)

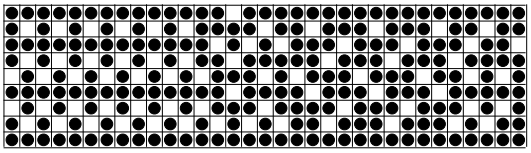


Figure 8: A Garden of Eden pattern

Theorem 1 (Moore-Myhill [9, 10]) *A cellular automaton is injective (one-to-one) on finite configurations if and only if it is surjective (onto).*

We do not give here the proof of this theorem. It is based on a combinatorial argument.

Theorem 2 *Life has a Garden of Eden configuration. Equivalently, Life is not surjective.*

Proof. Let us consider two finite configurations; in the first one, every cell is dead, and in the second one, only one cell is alive. According to the rules of the game, the living cell will die at the next time step. Hence, two different finite configurations have the same image: the cellular automaton is not injective on them. From Theorem 1, it is not surjective either: there exists a configuration which cannot be reached. □

Furthermore, as the set of finite configurations is dense in the set of all configurations for the product topology, one can prove the following: there exists a finite part of configuration (beware that it is sometimes called a pattern) that cannot be found in a configuration after one iteration of life (see also Figure 8).

For those readers who are interested in global properties of cellular automaton, see the review paper [4].

This study of Garden of Eden configurations for life was motivated by the following notion (see [16, 3]):

Definition 2 (von Neumann) *A cellular automaton is* construction-universal *for a set of configurations $\mathcal{A}$ if and only if any configuration of $\mathcal{A}$ is the image of a configuration.*

Remark that when no set is mentioned, the considered set of configurations is the set of *all* configurations. In this case the construction-universality is equivalent to surjectivity.

Corollary 1 (of Theorem 2) *Life is not construction-universal.*

3 Boolean circuits simulation

There is no theoretical problems in defining this notion: a device is universal for boolean circuits if it can simulate any boolean circuit. Any reasonable definition of the notion of simulation is convenient for our purpose; beware that it will not be the case for Turing-universality and intrinsic universality. The construction imitating a "real" machine is described in details in [2]. Here, we only present the ideas and patterns used to show Life's universality for Boolean circuits.

3.1 Wires and pulses

In a real machine, there are

1. *pulses* of electricity,
2. *wires* along which pulses of electricity go,
3. a *clock* generating pulses at regular intervals,
4. *logical gates* AND, OR and NOT,
5. auxiliary storage *registers*, each of which will store an arbitrarily large number.

In Life,

1. *pulses* are represented by *gliders*,

2. *wires* are represented by certain lines in the plane along which gliders travel (because they travel diagonally, we turn the plane through 45°, so they move across, or up and down, the page in Figures 12, 13 and 14),

3. the rule of the *clock* is played by a *Glider Gun*,

4. patterns imitating *logical gates* are presented in Section 3.2.

3.2 Logical gates

Let us explain now how logical gates can be simulated. As there are patterns (such as gliders or spaceships) crossing the plane, they can meet each other. Depending on their relative positions, many things can happen. Here, we present three reactions that will take part in the construction of the gates.

Gliders' crash. When two gliders meet each other, they can have, for instance, a vanishing reaction. Here, we only show one (the fastest) of the many possible ways to get such a reaction (see Figure 9) that we call a *crash*.

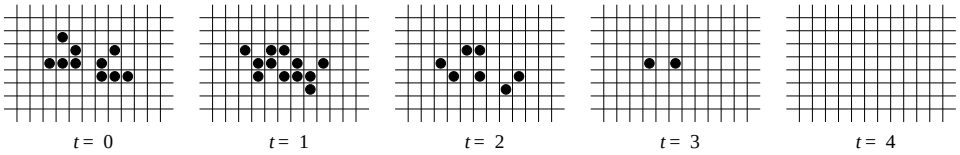


Figure 9: Gliders crash

Glider meets eater. Two gliders can also give birth to an eater —defined in Figure 3. Then, an eater, as its name shows, can eat many other patterns. For instance, a glider is eaten in Figure 10.

The kickback reaction. Another interesting reaction is the kickback shown in Figure 11. A glider can meet another one in such a way that one of them dies and the other one continues his way but in the opposite direction.

With the help of these reactions, the logical gates NOT, AND and OR are constructed as shown in Figures 12a, b and c, respectively. The crash reaction is featured with a grey hexagon. A and B are information: a *glider* (for 1) or *nothing* (for 0).

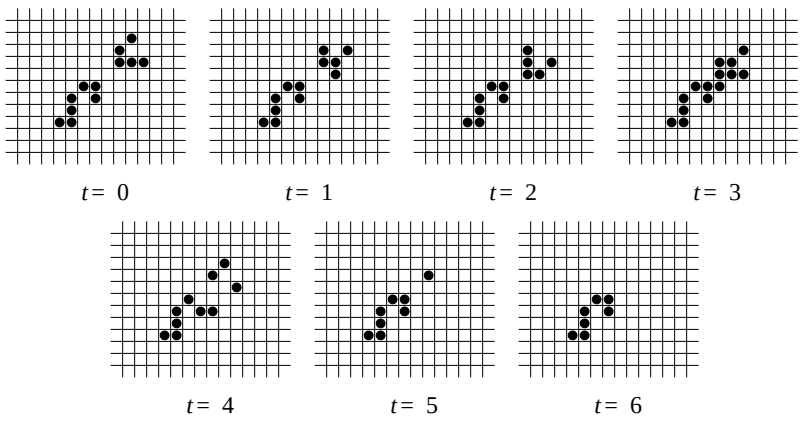


Figure 10: Glider is consummated by an eater

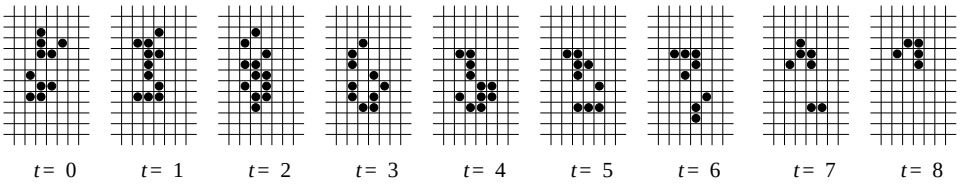


Figure 11: Kickback.

NOT **gate.** If A is a glider, then it has a crash reaction with a glider emitted by the gun and none of them survives. Hence, the result is nothing, that is, $\overline{A}$.

If A is nothing, then the glider emitted by the gun can continue its way, hence the result is a glider, that is $\overline{A}$.

AND **gate.** The result is a glider if and only if, after the second crash, there is a glider continuing its way from B's direction: B has to be a glider and nothing can arrive from the Gun's direction. It is possible if and only if the glider emitted by the Gun is killed in a crash reaction by a glider: A has to be a glider ($(A \text{ and } B) = 1 \iff A = 1 \text{ and } B = 1$). If none of A and B is a glider, then the glider emitted by the Gun can walk on the plane even infinitely. An eater is placed after the second crash in order to avoid this problem.

OR **gate.** The result is not a glider if and only if, the glider emitted by the top gun is killed by a glider coming from the direction of the bottom gun. It is possible if and only if this glider survives the two crash

reactions, that is, if and only if neither A nor B is a glider ((A or B) $= 0 \iff A = 0$ and $B = 0$). For the same reason as above, an eater is placed at the issue of the first crash reaction.

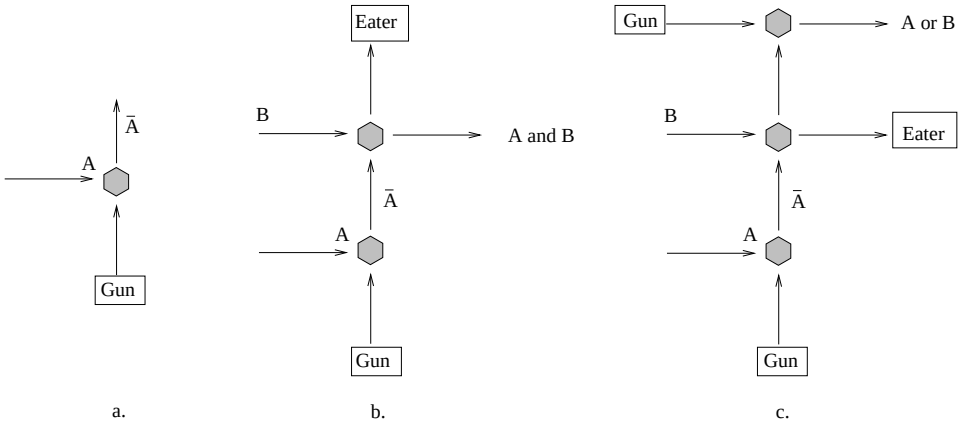


Figure 12: Logical gates NOT, AND and OR

Unfortunately, there is a problem: (A AND B) and (A OR B) are parallel to the input information A and B, while the NOT gate turns the information through 90°.

We have now to find a way to turn an information without complementing it, or, to complement it without turning it (the two problems are equivalent). We show how to realize the second operation.

The NOT gate without turning information. The glider streams that emerge from normal guns are so dense that they cannot collide without interfering with the following gliders. Before constructing a NOT gate without turning information, we show how to sparse a glider stream (see Figure 13). Two guns emit gliders in parallel paths but in opposite directions in such a way that a glider walking perpendicularly to these paths has a kickback reaction with gliders emitted by these guns. If the distance between these paths is $n/2$, then every nth glider is killed in both streams. Then, as shown in Figure 13, through these holes, every nth glider emitted by a third gun in a parallel path to the "walking" glider can cross without interacting. To get the phasing right, n must be divisible by 4, but it can be arbitrarily large and so we can make an arbitrarily thin stream.

We suppose that the gun emits gliders every 120 generations ($n = 4$). Then, when making a kickback reaction with the first glider, it is kicked

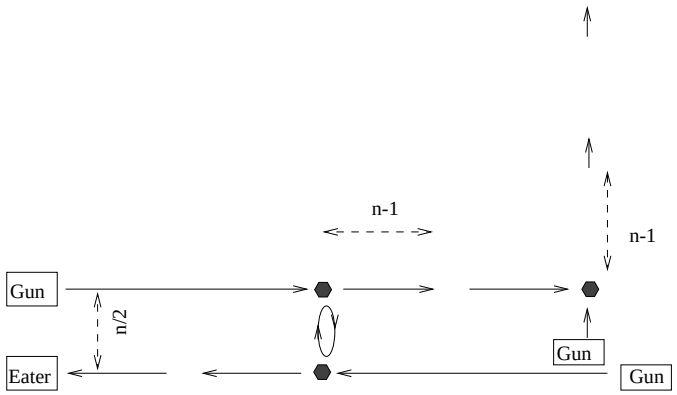


Figure 13: Sparsing a glider stream

back, the second glider crashes into the first one, forming a block and the third glider annihilates the block.

Let us see now how to make copies and complement of an information without turning it. See Figure 14.

We suppose that an information A is present every ten place ($000000000A$): A can be a glider ($A = g$) or nothing ($A = 0$). We consider that A is the first "emitted" information, followed by nine 0. When a stream $000000000A$ feeds into an OR gate with a stream being empty except at the second place where there is a glider g ($00000000g0$), the stream leaving the OR gate is the information A followed by a glider ($00000000gA$). Let us denote by X987654321 a stream of 10 gliders emitted by a gun (as described above), that we call a *full stream*. When the stream $00000000gA$ has a kickback reaction with a full stream, a copy of the information is reserved at the second place ($00000000A0$):

- if A was a glider ($A = g$), it has a kickback reaction with the first glider of the full stream. As described above, gliders 1-3 of the full stream are killed and X987654000 continues its way (as it is presented under the horizontal flash starting from the first crash reaction). As the second glider of the full stream is killed, $A = g$ can continue its way and the stream leaving the crash is $00000000g0$: in Figure 14, this case is represented at the left-hand-side of the flash ascending from the first crash.

- if A was not a glider ($A = 0$), the first glider of the full stream continues its way, and gliders 2-4 are dead, killed by the glider arriving at the

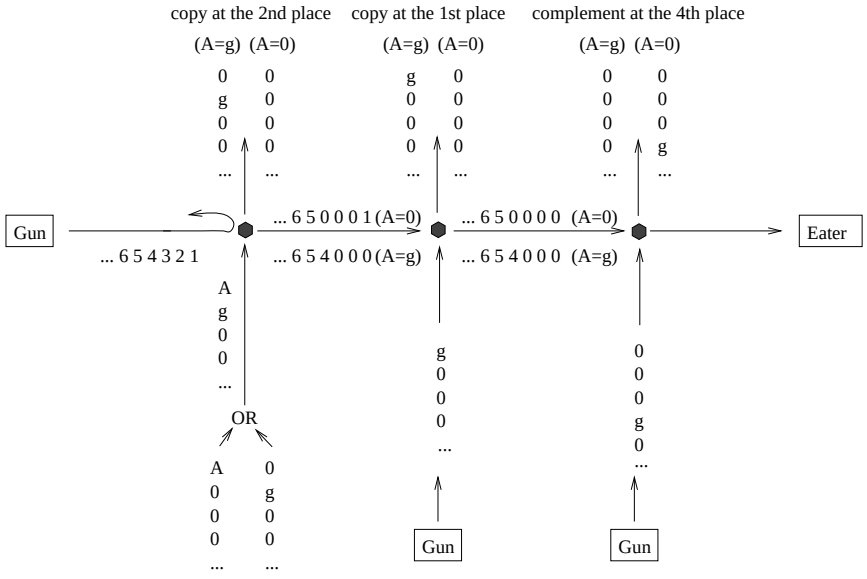


Figure 14: Copies and complement without turning information

second place: X987650001 continues its way (as it is presented on the horizontal flash starting from the first crash reaction). Then, after the crash reaction, the result is a stream without any glider, that is, 0000000000, as it is shown at the right-hand-side of the flash ascending from the first crash.

Using other crash reactions, we can reproduce another copy of A (crashing the stream ...654000/...650001 to a stream with only one glider at the first place $000000000g$) or the complement of A (crashing the stream ...654000/...650000 to a stream with only one glider at the fourth place $000000g000$), as shown in the figure.

3.3 The circuit

Now let us organize all these devices to obtain a boolean circuit. First we use wires but we also need that the pertinent information arrive at the same time on the input of the boolean circuit. Thus we need to introduce delays which is not difficult by turning the wires (see Figure 15).

Thus we obtain a circuit. It is enough for our simulation and we can say that Life is universal for boolean circuits.

Now remark that because of the small period of the glider guns, only

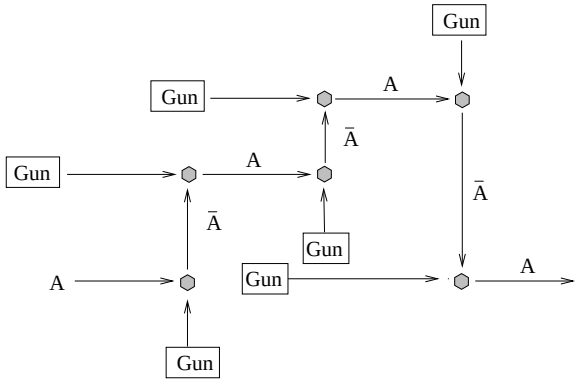


Figure 15: Delays

some information received at the end of the circuit is pertinent: the outputs at times corresponding to the traversal length of the circuit. This is not a problem in our case: we can ignore them! It becomes a serious problem for Turing-universality since in this case a 2 Registers Machine is simulated: these parasitic outputs can induce some extra modifications on registers.

In order to avoid these parasitic outputs, it is sufficient to increase the period of the glider guns and to adjust this period to the traversal length of the circuit. Alas this transformation by itself increases traversal length of the circuit because these new glider guns are much larger. Furthermore, in Conway's construction the size of such glider guns is proportional to the period (see Figure 13). In this particular point lies the error of Conway's construction — as far as we understood it.

This problem can be solved by 3 methods. The first one and the most general would be to find an infinite family of recursively defined glider guns so that their size is little "o" of their period. We were told that this construction exists but could not get any reference of it. If some reader has some information about on such a family, we would be glad if he could pass it onto us. With the help of this family, it would be possible to use glider guns of period larger than the traversal length of the circuit, hence correct the original proof.

The second method is to remark that there exists a small universal 2 Registers Machine machine and that it is sufficient to simulate it (and not any 2 Registers Machine). In this case, we think that among all known glider guns (many of them can be found on the WEB), one can find an adequate glider gun for using in this universal 2 Registers Machine simulation. We agree that this does not constitute a formal proof!

The last method that we propose is to use intrinsic universality of Section 5 because, in this case, this problem of modifying a register does not arise; thus, it is not needed to suppress parasitic outputs. The simulation can be realized using a fixed period glider gun. The period of this gun should not be too small not to create jams in logical gates, but it is the only constraint that should be observed — we can use a *large* gun.

4 Turing universality

4.1 Definition and discussion

Let us first explain why a precise definition of a "Turing-universal cellular automaton" is needed. In the case of standard computation models (e.g. Turing machines) an input word is transformed into an initial configuration which is a finite object. In that case, the natural requirement is that this transformation is recursive. In the case of a cellular automaton, the initial configuration is infinite and the cellular automaton acts on the whole configuration hence we have to explain what kinds of transformations from finite words to infinite (initial) configurations are allowed. The standard restriction to recursive transformations is not meaningful since this transformation maps a word onto an infinite object.

Recall that in order to define completely the computation model associated to a cellular automaton, we need to define

- the already mentioned transformation from a word to an initial configuration (*the encoding*),
- *the halting condition,*
- *the decoding*, *i.e.* the transformation forms the configuration to the output word. (As often, this is not a problematic point).

The first idea is to define the encoding as a transformation from words to finite configurations of the cellular automaton *i.e.* to almost everywhere quiescent configurations). The first problem is that all cellular automata do not have quiescent states; second and more important: in many simulations a periodic media —*i.e.* a periodic background configuration— is needed at infinity.

Thus the second idea is to transform an input word w into a program of a total recursive function f_w that defines the initial configuration. This

total recursive function would map the set of cells to the states. Unfortunately, this elegant definition is not convenient for us: consider the following encoding function:

$$f_w(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ w_i & \text{if } 1 \leq x \leq |w|, \\ 0 & \text{if } x = |w| + k \text{ where } k \in \mathbb{N}, \text{ and not } U(w,k), \\ 1 & \text{if } x = |w| + k \text{ where } k \in \mathbb{N}, \text{ and } U(w,k). \end{cases}$$

where $U(w,k)$ is true if and only if the universal Turing machine halts on the input word w in exactly k steps. The mapping $w \mapsto f_w$ is recursive but if we accept this encoding it contains in itself all the "computation" of the model and in this case, a trivial cellular automaton as the *shift* is universal. This is not acceptable thus our definition should be more restrictive.

We propose to consider the following definition of acceptable encodings. Such an encoding should map a word w onto an almost everywhere periodic configuration. A periodic configuration is formed by the repetition of a finite pattern under the "natural" transformations of the space of cells. For instance in 3D the configuration should be the repetition of a cube *i.e.* invariant under 3 independent translations. The non-periodic part of the initial configuration should be obtained recursively from w (and thus its size is bounded by a recursive function of w).

We are not sure that this definition is the most general. But all the simulations we met in the literature fit these requirements.

Concerning the halting condition, the first idea is that a "special" state should appear somewhere in the configuration (see [7]). This is not good because a new specially dedicated state should be used. For instance in Life the halting condition is different. We could also ask that the dynamics of the cellular automaton becomes stable (*i.e.* enters a time-periodic loop). This definition seems too restrictive as it excludes the billiard models (see [15]). We propose a more general notion: the halting condition is a total recursive function from the non-periodic part of the current configuration to the pair $\{\text{halt}, \text{continue}\}$.

It is natural to consider each iteration of the considered cellular automaton as a computation step.

In this context, the requirements for the decoding function are also natural: this function should take into account the non-periodic part of the current configuration, and map it recursively to a word.

4.2 Construction of a two Register Machine

In order to define a pattern imitating a real machine, in addition to the devices constructed for boolean circuit's simulation, we also need some auxiliary storage units. In [2], Conway chooses to simulate 2-Registers Machines (2RM for short) that are Turing-universal [8]. The construction of registers —and operations on them— are also described. In brief, a block is used to represent a static piece of information, two gliders are used to pull a block back three diagonal places and ten gliders are used to move up a block just one diagonal place. The precise construction is very complex.

5 Intrinsic universality

5.1 Definitions and discussion

Although the intuitive notion of simulation between cellular automata is broadly used, it does not always correspond to the same definition. For instance, in some cases the number of steps used to simulate one step of the considered cellular automaton may vary. Another difference is that sometimes it is not the same function that is used to encode and decode the configurations.

We present below a rather strict definition of this intuitive notion inspired by the notion of simulation in [14]. We do not know what is the most general definition that could be accepted.

Definition 3 *Let $\mathcal{A}$ be a cellular automaton and $C_{\mathcal{A}}$ the set of its configurations. Let $\mathcal{B}$ be a cellular automaton and $C_{\mathcal{B}}$ the set of its configurations. We say that $\mathcal{B}$* simulates *$\mathcal{A}$, if there exist a one-to-one (injective) application $f : C_{\mathcal{A}} \to C_{\mathcal{B}}$ UL-defined (see below) and a constant T in $\mathbb{N}$ such that for all c in $C_{\mathcal{A}}$*

$$f(\mathcal{A}(c)) = \mathcal{B}^T(f(c)).$$

T is the simulation time factor, that is, the time which is necessary for $\mathcal{B}$ to simulate one iteration of $\mathcal{A}$. It depends on f but not on c. f is called the *encoding function;* "f is UL-defined" means that f is defined uniformly and locally. This notion is not completely defined here because our aim is that it is true on any space on which we could imagine a definition of cellular automaton. This notion depends on the definition of cellular automaton we use. In the simple case of 2D cellular automaton, f is UL-defined means that it commutes with 2 independent shifts. This definition can be illustrated by the following diagram:

$$\begin{array}{ccc} c & \xrightarrow{f} & f(c) \\ \mathcal{A}\Big\Downarrow & & \Big\Downarrow\mathcal{B}^T \\ \mathcal{A}(c) & \xrightarrow{f} & \mathcal{B}^T(f(c)) \end{array}$$

Remark that if we endow "naturally" $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ with the product topology, f is a continuous function that is invariant under n independent shifts where n is the dimension of the space. From [6, 12] we deduce that, basically, f is a cellular automaton with, in addition, the possibility to take into account a finite information concerning the position of the cell. This finite information must be periodically distributed on the space.

The differences between our definition and the definition of [14] is that we add the requirement that f is UL-defined. Our goal is to make definition of simulation compatible with Turing universality: if a cellular automaton $\mathcal{A}$ is simulated by a Turing-universal cellular automaton $\mathcal{B}$, then $\mathcal{A}$ is universal and conversely.

But, this definition does not cover all simulations. In [13] the problem of simulating cellular automata with only one-way communication between cells has been studied: "given a Cayley graph, can all bidirectional cellular automata be simulated by a one-way cellular automaton on this graph?". Sometimes, such a simulation is possible, but after the simulation of each iteration of the bidirectional cellular automaton, the obtained configuration is shifted in the one-way cellular automaton.

Beware that a strict interpretation of the term "intrinsic universality" would mean that we simulate any other cellular automaton defined on the 2D grid, but not on an hexagonal lattice. Our result is a little stronger since we simulate any 2D cellular automaton. It is stronger because any cellular automaton —e.g. on a hexagonal lattice— can be simulated by a 2D cellular automaton (and even with Von Neumann neighborhood). This is also true for any "regular" lattice of the plane, as proved in [14].

5.2 Our construction

We recall that one can define patterns imitating

- pulses and
- logical gates.

As the NOT gate can be imitated with and without turning information, one can also turn information as often as one wishes. Hence,

- wires of arbitrary length can be constructed.

Let us consider any 2D cellular automaton $\mathcal{A}$ and let us define first our encoding function f. In the context of our definition, the cellular automaton $\mathcal{B}$ is Life. We shall explain what is the simulation time T at the end of this section.

Let us first consider the local transition rule of $\mathcal{A}$ that we call δ. If we encode states (for instance in binary) by words on the alphabet $\{0,1\}$, δ can be seen as a boolean function. We saw in Section 3 that we can simulate with our cellular automaton any boolean function via the simulation of wires, logic gates, crossovers, fanouts, delays, etc. Thus we can construct the circuit corresponding with δ in a rectangle (see Figure 16).

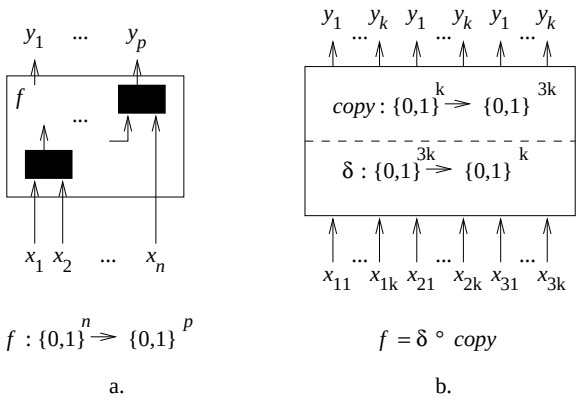


Figure 16: The simulation of δ

The idea of our simulation is to transform each cell into large squares containing this rectangle and the neighborhood connections between cells (see Figure 18). For simulating 1D cellular automata, we obtain the simple Figure 17 (output arrows represent copies of the same information).

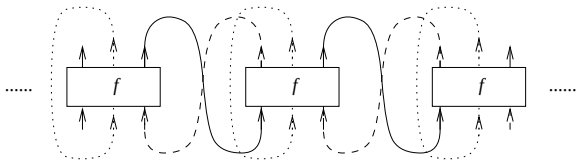


Figure 17: Simulation of 1D cellular automata

The state of each cell is naturally encoded by the corresponding word (in binary) on the output wires of the associated rectangle.

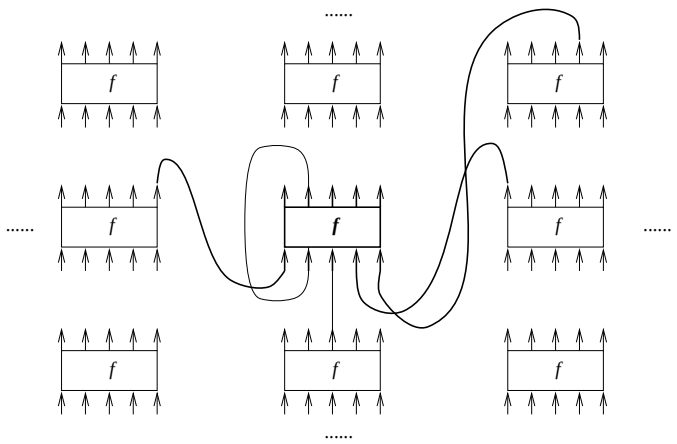


Figure 18: The general scheme

Now, let us examine the problem of connections between the cells. A neighborhood connection should be represented by a bunch of wires starting from the output of the rectangle associated with one cell to the input of the rectangle associated with the other one.

A synchronization is needed between all these wires to ensure that the informations arrive simultaneously from all neighbors. It is straightforward using delays defined in Section 3.

Theorem 3 *For all neighborhood there exist a sufficiently large size of square such that the simulation is possible,* i.e. *such that it is possible to draw all needed wires, delays, and crossover between cells.*

Proof. The idea of the proof is that we can zoom the general scheme of computations in order to introduce those necessary delays.

First, let us consider the general scheme of the construction in which wires are represented by curves (see Figure 18).

It is not difficult to see that we can embed this scheme into a configuration of our cellular automaton; the only problem is how to solve the synchronization constraints. We propose the following procedure:

- compute the length of all input wires,
- consider the largest difference between these lengths. Let n be such a difference.

Then we need a square of size $\mathcal{O}(\sqrt{n})$. If we have no space to put the necessary delay, we know that there exist a scale factor M such that we can include in our scheme the necessary delays making a $\times M$ zooming because, by making this zooming, the delay that we introduce is proportional to M^2, while the needed delay n grows linearly in M.

Let us now continue the procedure:

- zoom $\times M$, and put the needed even delays in the rectangle.
- if a delay is odd: move that cell of one unit, such that the corresponding delay becomes even.

To avoid the problem that the signals must arrive separated to the crossovers, we introduce a delay before one of the inputs and another one after the other output and then zoom again accordingly.

The time T of the simulation is obtained by these successive zoomings. □

6 Open problems

The definition of simulation between cellular automata presented in Section 3 is maybe not the most general one. We think that this notion is worth studying.

Another problematic is to find the smallest Turing-universal or intrinsically universal cellular automaton: it is interesting to know whether small cellular automata can be universal in order to construct them with very elementary natural devices and observe complicated behaviors. Contrarily to Turing Machines, cellular automata are canonically defined: no problem to know whether heads of Turing machines could be allowed to stay or just go left or right, no problem of semi-infinite or bi-infinite tape, no problem to know whether the initial state can or cannot be used after the first step ... This problem of finding the smallest universal cellular automata is much more interesting than for Turing Machines.

Even if it is conjectured that 2 states and 3 neighbors are sufficient (see [17]), for Turing universality, the smallest universal cellular automaton that we know (both Turing and intrinsic universalities) is presented in [5].

References

[1] E. R. Banks. *Information processing and transmission in cellular automata.* PhD thesis, Mass. Inst. of Tech., Cambridge, Mass, 1971.

[2] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for your mathematical plays*, volume 2. Academic Press, 1982.

[3] A. W. Burks. *Essays on Cellular Automata.* University of Illinois Press, 1970.

[4] B. Durand. Global properties of cellular automata. In E. Goles and S. Martinez, editors, *Cellular Automata and Complex Systems.* Kluver, 1998.

[5] A. Gajardo, B. Durand, and E. Goles. Universality in a 2-dimensional cellular space with a neighborhood of cardinality 3. *submitted*, 1997.

[6] G. A. Hedlund. Endomorphism and automorphism of the shift dynamical system. *Mathematical System Theory*, 3:320–375, 1969.

[7] A. R. Smith III. Real-time language recognition by one-dimensional cellular automata. *Journal of Computer and System Sciences*, 6:233–253, 1971.

[8] M. L. Minsky. *Computation: Finite and infinite machines.* Prentice-Hall, 1967.

[9] E.F. Moore. Machine models of self–reproduction. *Proc. Symp. Apl. Math.*, 14:13–33, 1962.

[10] J. Myhill. The converse to Moore's garden-of-eden theorem. *Proc. Am. Math. Soc.*, 14:685–686, 1963.

[11] F. Nourai and S. Kashef. A universal four states cellular computer. *IEEE Transaction on Computers*, C24(8):766–776, 1975.

[12] D. Richardson. Tesselations with local transformations. *Journal of Computer and System Sciences*, 6:373–388, 1972.

[13] Zs. Róka. One-way cellular automata on Cayley graphs. *Theoretical Computer Science*, 132:259–290, 1994.

[14] Zs. Róka. Simulations between cellular automata on Cayley graphs. *Theoretical Computer Science*, 1998 (to appear).

[15] T. Serizawa. Three state neumann neighbor cellular capable of constructing self reproducing machines. *System and Computers in Japan*, 18(4):33–40, 1987.

[16] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[17] S. Wolfram. Universality and complexity in cellular automata. In T. Toffoli D. Farmer and S. Wolfram, editors, *Cellular Automata*, pages 1–36. North-Holland, 1983.