

## Chapter 1

### Circuit Universality of Two Dimensional Cellular Automata: a Review

Anahí Gajardo\*

*Departamento de Ingeniería Matemática, Universidad de Concepción,  
Casilla 160-C, Concepción, Chile.  
anahi@ing-mat.udec.cl*

Eric Goles†

*Universidad Adolfo Ibáñez,  
Av. Diagonal Las Torres 2640, Peñalolén, Santiago, Chile.  
and  
Instituto de Sistemas Complejos de Valparaíso (ISCV),  
Av. Artillería 600B, C° Artillería, Valparaíso, Chile.  
eric.chacc@uai.cl*

Universality of Cellular Automata (CA) is the ability to develop arbitrary computations, and is viewed as a “complexity certificate”. The concept exists since the creation of CA by John von Neumann, and it has undergone several transformations and ramifications. We review a sample of models, starting with Banks’s CA, where universality has been shown through the construction of arbitrary boolean circuits (“Circuit Universality”), in most but not all cases leading to proofs of Turing Universality.

#### 1.1. Introduction

A  $d$ -dimensional Cellular Automata ( $d$ -CA) is a dynamical system evolving in  $\mathbb{Z}^d$  in discrete time, where the upgrade of the lattice is synchronous and each site changes its state following a local rule which depends on the states of a fixed neighborhood.

\*This work was partially supported by FONDECYT #1061036.

†The author is also affiliated to the Centro de Modelamiento Matemático (CMM), Universidad de Chile, UMR 2071-CNRS. This work was partially supported by FONDECYT #1070022.

Such a system can present very diverse and complex behaviors, the prediction of which is usually difficult. But what is exactly meant by this? If we completely know the initial configuration of a CA, we can *compute* its whole evolution up to any iteration  $t$ . On the other hand, if we only know the state of a finite part of the lattice, we can only update the state of those cells whose neighbors's states are all known. In this case the set of cells that can be updated decreases over time until an instant in which we cannot compute anymore.

Let us illustrate this with an example in  $\mathbb{Z}$ . Consider the following rule: if the central cell is in state 1, it will change to 0 if any of its neighbors is in state 0; in any other case, it keeps its current state. (Figure 1.1 illustrates the evolution of this rule for a certain initial configuration). Let us now suppose that we know the initial state of cells 1 to 10. Initially, we can update only cells 2 to 9. In the next iteration we can update only cells 3 to 8, and, in general, at iteration  $i$  we can only update cells  $1 + i$  to  $10 - i$ . At step 5, the state of every cell is unknown to us.



Fig. 1.1. A space-time diagram of a 1-dimensional Cellular Automaton. Time evolves upward.

In general, in a  $d$ -dimensional CA with a neighborhood of radius 1, if we know the state of a hypercube of side  $2t$ , we can compute the state of the central cell for only  $t - 1$  iterations; the computation will take  $O(t^{d+1})$  operations in a serial computer.

What means *to predict* in this context? It means to compute faster than that. In this context the previous CA is predictable: we can assert that the state of the central cell will be 1 at iteration  $t - 1$  if and only if we see no cell in state 0 at the beginning. Computing this last proposition takes only  $O(t^d)$  on a serial computer, and  $O(1)$  on a parallel computer with  $O(t^d)$  processors. Thus, we can say that this CA is *simple*.

One may define the complexity of a CA rule  $f$  as the complexity of the following problem:

**(CA-VALUE( $f$ ))<sup>a</sup>** Given the configuration of a finite hypercube

<sup>a</sup>this problem has been also called "CA-PREDICTION" by C. Moore.

$c$  of size  $2t$  and a given state  $s$ , decide whether after  $t-1$  steps,  $s$  will or will not be the state of the central cell of the hypercube, when the initial configuration is  $c$  and the CA rule is  $f$ .

Now, let us consider the class of polynomial problems (P), *i.e.*, problems which are solvable in polynomial time on a serial computer. Clearly the problem of predicting the state of a site at step  $t$  in a CA belongs to P. Within the class P, an important subclass is the class NC of problems that can be solved in polylogarithmic time on a parallel computer with a polynomial number of processors. It is not known whether  $P=NC$  or not; like in the case of the  $P=NP$  question, the concept of P-completeness plays an important role. P-complete problems are such that if any of them belongs to NC, then the complete class P is in NC, because every polynomial problem *reduces* to every P-complete problem. If, as is widely suspected,  $P \neq NC$ , then being P-complete probably means to be *inherently secuencial*, out of NC. See R. Greenlaw *et al.*,<sup>1</sup> for example, for a detailed presentation of this subject.

An important P-complete problem<sup>2</sup> is CIRCUIT-VALUE:

**(CIRCUIT-VALUE)** Given a directed acyclic graph whose vertices have 0,1,2-ary logical gates, given a truth value assignment to each 0-ary vertex, and given a particular vertex called *output*, decide whether the output is or not True after computing each logical gate.

If we reduce CIRCUIT-VALUE to CA-VALUE( $f$ ), we prove that the second problem is P-complete. To achieve that for a particular 2-CA, it is enough to embed circuits in the lattice. In this case, the automaton is said to be *Circuit Universal*.

The technique of emulating a boolean circuit by a CA was first used by E. R. Banks<sup>3</sup> (1971). Banks embedded a complete computer design in the cellular lattice in order to prove that the CA was *Turing Universal*, *i.e.*, that it was able to simulate a Universal Turing machine (TM). Turing Universality implies Circuit Universality, since with a TM we can compute a circuit. The converse is not necessarily true, but it often turns out to work: in the case of Banks, his method to simulate circuits also allowed the simulation of a Turing machine, because a Turing machine can be computed with an infinite periodical circuit. See Z. Róka and B. Durand,<sup>4</sup> N. Ollinger<sup>5</sup> and J-C. Delvenne *et al.*<sup>6</sup> for extensive discussions of these and other notions of universality in Cellular Automata.

Evidently, circuit simulation is not the unique path to proving Turing Universality. One dimensional cellular automata, for instance, do not lend themselves easily to building logical circuits (though, if they are Turing Universal, we know that they can compute them). In particular, M. Cook showed the Turing universality of the elementary CA with Wolfram's code 110 by simulating a TAG-system.<sup>7,8</sup>

In this paper we will review several Circuit Universal models taken from different domains, thus giving a sample of different techniques. Some of these models are well known and/or important, arising mostly in physics and the field of artificial life; other have been chosen mostly for their technical interest.

## 1.2. Computing through signals

Turing Universality of two dimensional cellular automata has been proved mainly by emulating logical gates, in particular when working with a small number of states.

Roughly speaking, the idea is to represent logical values as 'signals' that travel in the space and interact with each other. The interactions must be rich enough to reproduce all the logical gates. Usually a signal represents the logical value *True*, and the absence of a signal is the logical value *False*. Logical gates are stable or periodical configurations, *i.e.*, finite configurations that, under certain boundary conditions, remain unchanged when iterating the CA rule, and if they change, they do it in a periodical way.

It is not necessary to exhibit each of the logical gates, a finite but complete set being enough. For example, it suffices to emulate a NOT and an AND gate, together with some devices allowing to direct, duplicate (a FANOUT) and to cross signals (a CROSSOVER) (Figure 1.2 shows a XOR gate constructed by pasting the needed pieces together). The CROSSOVER can be computed with a planar logical circuit by using only NOT and OR gates, and hence if we can emulate a NOT and an OR gate (or an AND gate) we do not need to give the CROSSOVER explicitly.

On the other hand, if we have the CROSSOVER, the AND gate and the OR gate are enough (the NOT gate is not necessary) because the MONOTONE CIRCUIT VALUE problem -which consists in computing a circuit with no NOT gates- is also P-complete.<sup>9</sup>

With all the pieces in hand, a logical circuit is emulated by a configuration of the CA such that some cells represent the value of the input

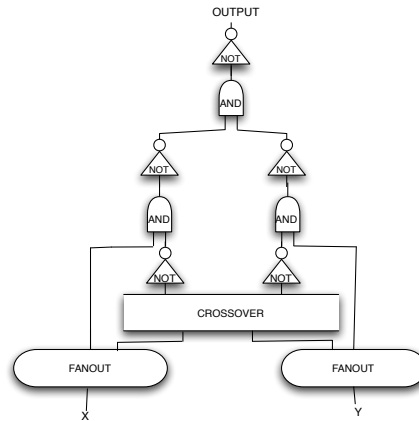


Fig. 1.2. A scheme which, embedded in a CA, computes a XOR gate.

variables. When the CA evolves, the signals advance, enter the logical gates, and the circuit is computed. After a certain delay, one can recover the result of the computation by looking at whether a signal appears or not as the output of the last logical gate. We remark that in order to work correctly, signals must be synchronized: all the trajectories from the inputs to the output must have the same length.

If the usual coding of True and False through presence and absence of signal is used, the NOT gate has some particular requirements. When a signal enters the NOT gate, nothing should exit; on the other hand, if no signal enters, the NOT gate must generate a signal. How will the NOT gate know when to generate this signal? This problem has been solved in two ways: 1) defining the NOT as a device that periodically emits signals when no signal enters; 2) allowing a second input to the NOT, which, in some sense, indicates when to compute.

The first solution has the inconvenient of periodically introducing signals in the system, thus producing many output signals before the “real” computed output. This is not as bad as it may sound, as long as we are able to discard all the spurious outputs and read only the good one. The second solution enlarges the amount of “wires” in the circuit, but only proportionally to the number of gates.

An additional feature may be wanted in the gates and devices, though it is not really necessary it is used to ensure that information flows in the

proper directions, *i.e.*, that gates do not send back signals in the direction of the inputs. This is achieved with a DIODE.

The first work that showed the universality of a CA by simulating a circuit was E. R. Banks's PhD thesis,<sup>3</sup> which defined the smallest two-dimensional universal automaton: a CA with two states and von Neumann neighborhood (*i.e.*, the four nearest cells in the 2-dimensional lattice). It is not easy to do this, and thus the Banks automaton with two states features some rather complicated devices. Banks also defined some three state automata, which are much simpler. In the next section we show one of them.

**1.2.1. A three states CA by Banks**

The local transition rule of Banks's CA is given in Figure 1.3. The figure shows the states of the cell together with its four nearest neighbors, followed by an arrow pointing to the state the cell will adopt in the next iteration. The rule is isotropic, and thus each configuration represents also its rotated versions. If a cell is in a situation not given in the list, then the cell does not change of state.

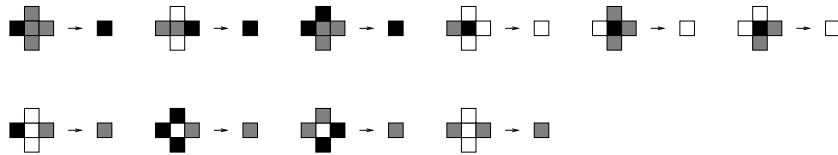


Fig. 1.3. The transition rule of the three states Banks's CA.<sup>c</sup>

Banks defined a wire: a line of cells in gray state, embedded in a background of cells in blank state. This is a stable configuration, but if two neighboring cells in the chain are changed to blank and black, this perturbation propagates over the chain in the direction of the black cell (this is a signal), and if two signals collide, they disappear; a cut wire is a dead end for the signal. A junction of three or four wires is also a stable configuration; a grey cell is added in the three wires case, giving it a short cut wire. When one or two signals (in right angle) enter a junction they exit by the remaining wires. But if three signals enter a junction, they disappear. Figure 1.4 shows simulations of some of these phenomena.

The OR gate is simply a junction of three wires, which can be also used as a FANOUT. These gates can send signals in the direction of the inputs,

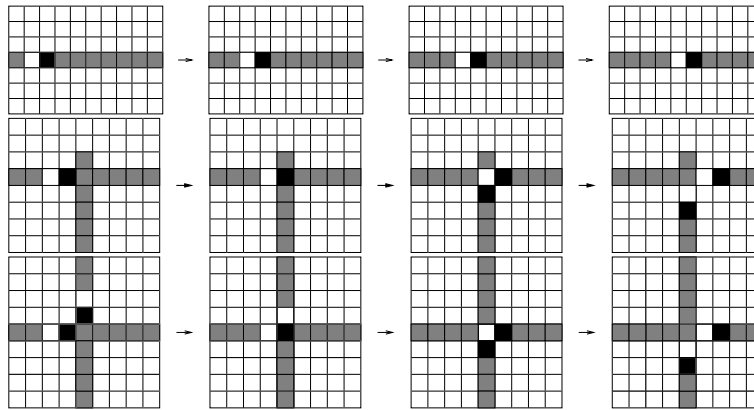


Fig. 1.4. Top: A wire and the propagation of a signal. Middle: When one signal enters a three wire junction. Bottom: When two signals enter a four wire junction.

something that Banks preferred to avoid. He managed this by defining a DIODE, shown in Figure 1.5(a), which only allows a signal to pass in one direction. Banks also constructed a *Timer* that generates signals periodically, shown in Figure 1.5(b): the signal inside the cycle duplicates each time it passes by the junction, sending a signal into the exiting wire. The NOT gate is composed by two Timers connected to a junction (see Figure 1.5(c)). If a signal A arrives to this junction at the same iteration that both Timer signals, they mutually annihilates.

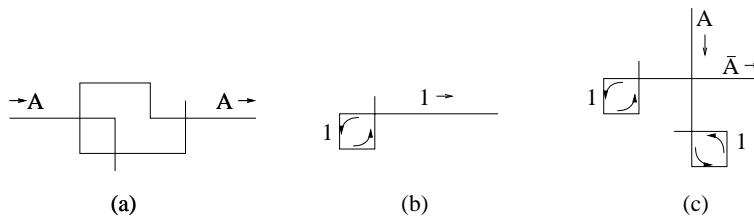


Fig. 1.5. (a) the Diode. (b) the Timer. (c) the NOT gate.

Another universal CA was defined by B. Silverman,<sup>10</sup> and is known as *Wireworld*. Its rule is very similar to those of Banks's CA, but it uses four states and Moore's neighborhood; this gives more flexibility and allows to build smaller and simpler circuits. In fact, a very small circuit that enu-

merates prime numbers has been constructed by D. Moore and M. Owen.<sup>11</sup>

### 1.3. CA over a hexagonal grid and three states

The rule of this CA<sup>12</sup> by the present authors is very similar to those of Banks's CA. The fundamental difference is the cellular space: it works over the hexagonal grid, with the cells located on the vertices of the grid, and having thus only three neighbors each. The transition rule is as follows:

- blank state remains blank except if it has at least a black and a grey cells in its neighborhood, in which case it becomes grey, it also becomes gray if it has exactly two gray and one blank neighbors,
- grey state remains grey except if it has exactly one black neighbor, in which case it becomes black,
- black state always changes to blank state, except if it has a black neighbor, in which case it becomes grey.

In this CA we can define the following two configurations, which are enough to construct all the needed gates, as we will show.

**Wire** : a connected chain of grey cells over a background of blank cells. This configuration is stable, but if two neighboring cells in the chain are changed to blank and black, this perturbation propagates over the chain in the direction of the black cell (this is a signal, Figure 1.6(a)), and if two signals collide, they disappear.

**Annihilating Junction** : a device with three connected wires, such that: if it receives one signal on one of the wires, it generates one signal at each of the two remaining wires; but if it receives more than one signal simultaneously, it absorbs them and generates no outgoing signal (Figure 1.6(b)).

The Wire ensures that we can transport the signals anywhere in the cellular plane. The Annihilating Junction acts as a FANOUT when it receives only one signal. It can also act as a XOR gate when two of its wires are used as inputs, and this XOR can be used to construct a NOT gate, by taking the idea of Banks. As in Banks's work, a DIODE is required if we want to prevent a return of the signals back to the inputs. Since Banks's DIODE uses the four wires junction, its construction cannot be imported here. Figure 1.7(a) shows the solution, composed by two successive NOT gates. They will not change the input signals, but if a signal enters by

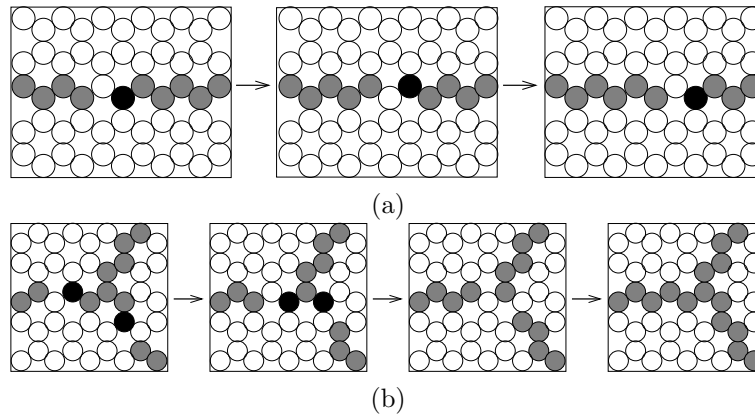


Fig. 1.6. (a) A signal and the wire. (b) Two signals entering an Annihilating Junction.

the right, a triple collision is produced in the output of the second NOT; nothing changes inside the DIODE, and no signal goes to the left.

Unfortunately, the XOR and the NOT gates are not enough for constructing the OR and the AND gates. In Banks's CA, the OR gate was obtained because two signals do not annihilate each other in a junction, something which does not work here. However, the logical gate ' $p$  AND NOT  $q$ ' can be computed within this system, as shown in Figure 1.7(b). In order to correctly use the gate, the input signals must be synchronized to collide at point C (or anywhere between the hexagon and the input  $p$ ). If a signal arrives by  $p$ , it will exit. If a signal arrives by  $q$ , it will die at the Annihilating Junction A. If two signals arrive at the same time they will be mutually annihilated at point C and no output will be produced. Any logical gate can be constructed by using the ' $p$  AND NOT  $q$ ' gate, and hence the cellular automaton is Circuit Universal.

**1.4. Life automata**

**1.4.1. Game of life**

John Conway's famous *Game of Life*,<sup>13</sup> which is known for its formidable variety of complex structures that make us think about life forms, is also universal. This automaton has two states (alive and dead) and Moore's neighborhood (*i.e.*, the eight nearest neighbors in the square grid). The rule is very simple: *a dead cell becomes alive if exactly three of its neighbors*

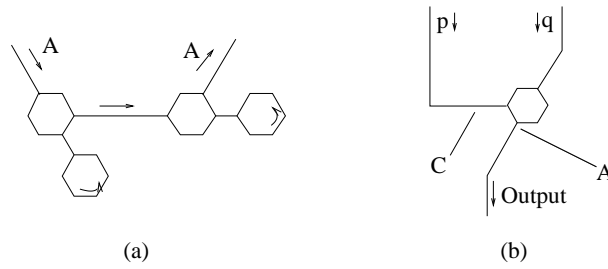


Fig. 1.7. (a) The DIODE. (b) The  $p$  AND NOT  $q$  gate.

are alive, and an alive cell remains alive only if two or three of its neighbors are alive.

Here the computation is made through signals that travel in the empty lattice, and the main logical gate is simply a collision. The Turing Universality of this automaton was proved by E. Berlekamp, J. H. Conway and R. Guy.<sup>13</sup> Here we will expose a variation of the proof by Z. Róka and B. Durand.<sup>4</sup>

The signals are called *gliders*, one of which is shown in Figure 1.8(a) while travelling in the lattice. Notice that its displacement is done in four steps, implying that there are several kinds of collisions, depending on the respective phase of the participating gliders. One of these collisions kills both gliders (see Figure 1.8(b)).

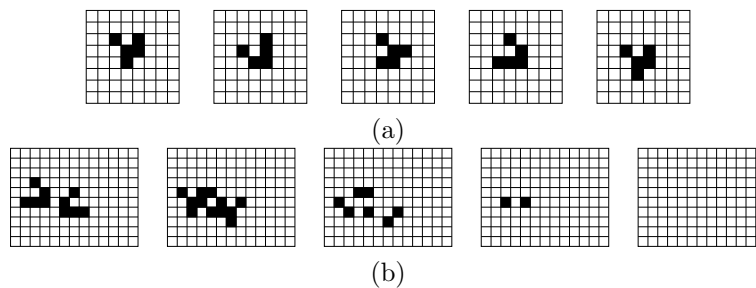


Fig. 1.8. (a) A glider traveling in the lattice. (b) A glider crash.

This collision emulates a gate with two outputs:  $p$  AND NOT  $q$  and NOT  $p$  AND  $q$ , because each glider will continue its way if and only if the other glider does not arrive. It is possible to kill one of the outputs with a

configuration called *eater*, which is a stable configuration that destroys the gliders when they collide with it. We obtain in this way a  $p$  AND NOT  $q$  gate. If we replace the input  $p$  by a periodical glider generator, which exists (a *glider gun*), we obtain a NOT gate. With these two gates all the other gates can be obtained.

To complete the construction, we still need to duplicate and to direct the gliders, and this cannot be done with a logical gate. To achieve this, another collision must be used: a collision called *Kickback Collision*. When a stream of gliders collides with a glider in a Kickback collision, the first glider comes back to the stream and collides with the second glider. This last collision can kill the third glider too. In short, a single glider can kill three. This can be used to duplicate and change the direction of a logical value, as shown in Figure 1.9. The FANOUT requires several gliders (1's), which can be generated with glider guns.

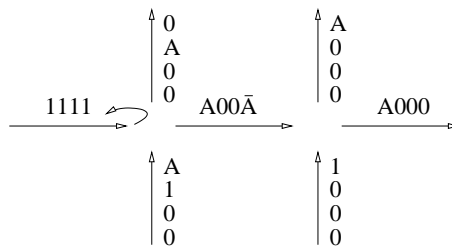


Fig. 1.9. A FANOUT. If  $A = 1$ , the first three 1's of the left are killed, and the 1's coming from the bottom survive, as well as the last 1 which finally exits by the right. If  $A = 0$ , the last three 1's of the left are killed, and no glider goes up; only the first one goes on to kill the 1 coming from the right-bottom, preventing the exit of gliders in any direction.

#### 1.4.2. Life without death

This CA has almost the same rule as the Game of Life, but in this case no cell ever dies. Here there are signals that travel in the lattice too, but they leave, of course, a trace of living cells. This phenomenon is called a *ladder*. The universality of this automaton was proved by D. Griffeath and C. Moore.<sup>14</sup> In their work, they establish three basic properties that the CA satisfies and that are enough to prove its P-completeness.

**ladder** : it is a configuration that grows in a straight line.

**L** : it is a stable configuration such that the ladder turns when colliding with it.

**ladder collision** : if one ladder collides with the trace of another ladder it stops.

With the collision we have a  $p$  AND NOT  $q$  gate and all the other gates, including the CROSSOVER. If a ladder turns three times it collides with its own trace and stops, and this can be used to stop unused outputs. The FANOUT can also be constructed with this property (see Figure 1.10). Since within this system we have no ‘ladder gun’, all 1’s must be produced at the beginning and delayed long enough to arrive at the desired iteration to the gate where they are needed. Griffeath and Moore proved that this can be done without making the circuit grow in an inconvenient way.

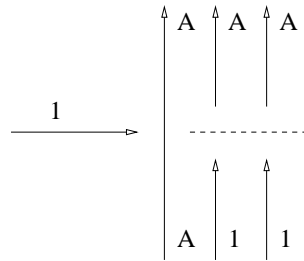


Fig. 1.10. If a signal  $A$  passes before the 1 of the right, every other 1 passes. But if the signal  $A$  is 0, then the 1 of the right prevents the other 1’s from going up.

### 1.5. Reversible models

Reversibility used to be seen as a limitation for universality. In fact, since the AND gate is not reversible, no reversible system can emulate an AND gate without producing some “garbage signals” (*i.e.*, additional outputs). For example, in the Game of Life, the basic gate  $p$  AND NOT  $q$  was obtained with a collision that generated two output signals and with an “eater” that killed unused output. In a Reversible Cellular Automata the equivalent of an “eater” cannot exist, and any “garbage signals” need to be “sent away” or “recycled.” In 1982 E. Fredkin and T. Toffoli<sup>15</sup> formally proved that any boolean Circuit can be computed with only wires and a given reversible logical gate now known as *Fredkin Gate*. Their construction uses a linear number of input constants and does not generate additional “garbage

signals.” Later, in 1990, K. Morita<sup>16</sup> showed that a universal reversible 1-CA can also be computed with only wires and the Fredkin gate, showing in this way that in order to have Turing Universality in a 2-CA it is enough to emulate wires and the Fredkin gate. Figure 1.11 shows the behavior of the Fredkin gate.

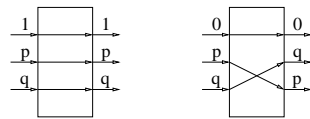


Fig. 1.11. The Fredking gate is a three-inputs-three-outputs logical gate. If the upper input value is 1 (True), the outputs are equal to the inputs. If the upper input value is 0 (False), the upper output is also 0 but the other two inputs are permuted.

Two important reversible models are: 1) the Billiard Ball Model<sup>15</sup>(BBM) (as well as its CA implementation<sup>17</sup>) and 2) the simulations of hydrodynamics and Navier-Stokes equations.<sup>18</sup> Essentially, both models represent a gas of identical hard spheres in a 2-dimensional space.

Let us start by studying the BBM. Figure 1.12 shows a balls collision. At time  $t$  we can put a ball either at  $X$ ,  $Y$ , or both. If we put both balls, they will collide following the outer paths. Otherwise, one of the other paths will be followed. Then, with this collision, we have a two-inputs-four-outputs gate, where the outputs are:  $X$  AND  $Y$  twice,  $X$  AND NOT  $Y$  and NOT  $X$  AND  $Y$ . With an ingenious construction (Figure 1.13) this gate can be used to compute a Fredkin gate.<sup>15</sup> Hence the general Billiard Ball Model is P-complete and Turing Universal.

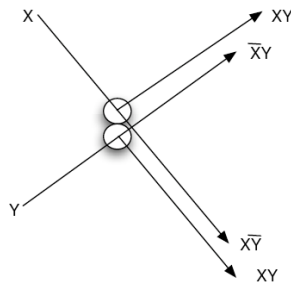


Fig. 1.12. The collision of two balls can be viewed as a general logical gate with multiple outputs. The variables  $X$  and  $Y$  represent the presence or the absence of a travelling ball.

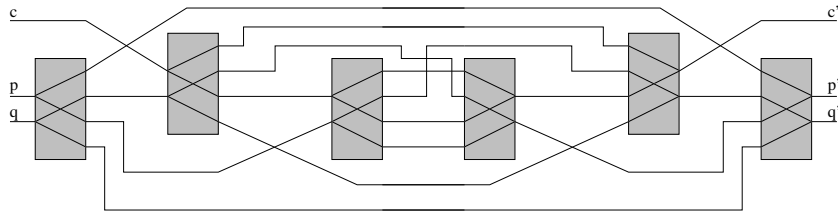


Fig. 1.13. The Fredkin gate computed only with the BBM collision. The gray boxes represent either the two-inputs-four-outputs gate given by the collision or its inverse. The lines outside these gates represent particle trajectories, which are controlled by well placed mirrors. The circuit must be constructed in such a way that the particles do not interact when their trajectories cross.

A sort of CA which simulates the BBM was defined by N. Margolus,<sup>17</sup> which he called *partitioned cellular automata* and have a special kind of iteration. Consider a 2-dimensional lattice with two states per site and divide the lattice in  $2 \times 2$  blocks of sites. A transformation will be applied to each block. In our case the transformation is given in Figure 1.14. It is conservative and reversible. The updating consists in applying the rule alternately to the  $2 \times 2$  blocks in the solid blocks partition and in the dotted one, as explained by Figure 1.15. One may see that this local rule allows to simulate the two dimensional BBM. Then, it is also Circuit Universal.<sup>17</sup>

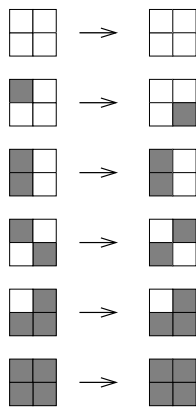


Fig. 1.14. Rules for the BBM partitioned automata

Other reversible CA, also called ‘partitioned CA’ but in a different sense,

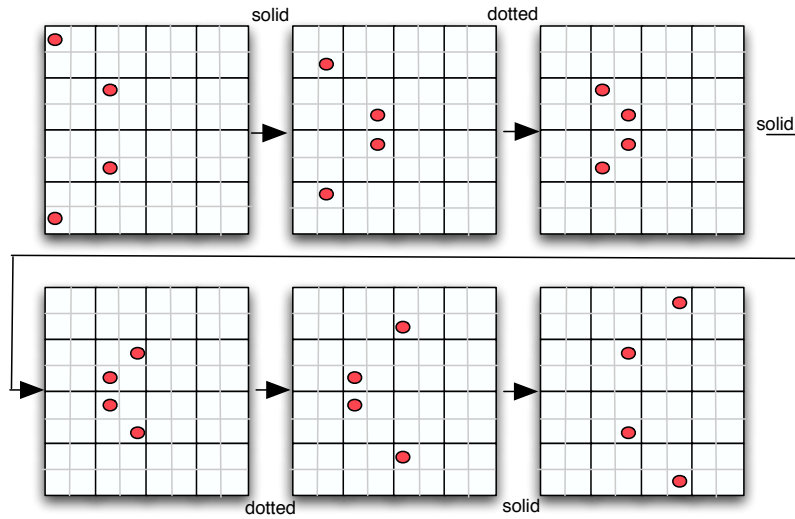


Fig. 1.15. Billiard ball collision in the BBM automata.

are those defined by Morita *et al.*<sup>19–21</sup> they are proper CA. Morita and his collaborators showed the universality of three simple 2-CA by emulating Fredking gates. Each publication cited does it for a different grid: a 32 states CA over the triangular grid, a 16 states CA over the square grid, and a 8 states CA over the hexagonal grid.

Lattice gas models are defined in a similar way as the BBM, also with collisions of particles. We will refer here to the simplest one, proposed by J. Hardy, O. de Pazzis and Y. Pomeau<sup>18</sup> in 1976. The HPP model considers only one kind of collision: when two particles collide head-on the outgoing particles are rotated by 90 degrees (see Figure 1.16). In any other case, the particles do not interact. This give us the same logical gate as the collision of the BBM, but here two of the outputs go back towards the inputs, requiring some tricky construction in order to recover them. The HPP lattice gas is P-complete, as was proved by C. Moore and M. Nordahl<sup>22</sup> in 1997. As an example, we give the construction of the NOT gate in Figure 1.17.

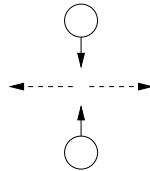


Fig. 1.16. Two ball head-on collision in the HHP lattice gas.

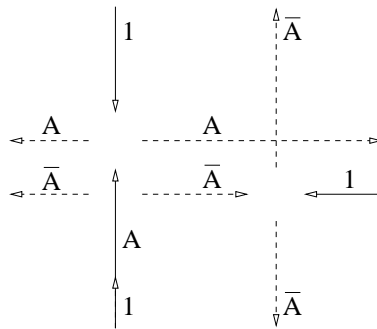


Fig. 1.17. NOT gate for the HHP lattice gas. If a signal comes by A, it will collide with the 1 coming from above and two horizontal signals will exit. If no signal comes by A, the 1 from below will collide with the other one, then a signal will exit and collide the 1 from the right to finally exit by the top.

### 1.6. Sandpiles

Bak *et al*<sup>23</sup> introduced a model, mostly known as the Sandpile Automaton, which captures important features of many nonlinear dissipative systems. Consider a set of sites in a  $d$ -dimensional lattice such that each cell is connected to the  $2d$  nearest neighbors. A finite number of tokens,  $x_i(t) \geq 0$ , is assigned to each cell  $i$ . Given the configuration at step  $t$ , if the site  $i$  has more than  $2d$  tokens, it gives one of these to each neighbor. Since the updating is synchronous, the site  $i$  also receives one token from each neighbor with more than  $d$  tokens. In Figure 1.18 we give an example of the Sandpile dynamics in a two dimensional lattice.

If the number of tokens is finite, within a finite amount of time every site will have less than  $d$  tokens (see, for example, C. Moore<sup>24</sup> for a formal proof). Then, any finite configuration becomes stationary (a fixed point), that is to say, every avalanche stops in a finite number of iterations. The

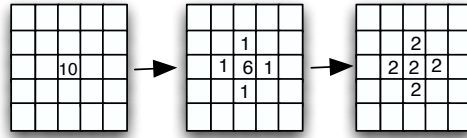


Fig. 1.18. A simulation of the Sandpile.

interesting thing is that before reaching the equilibrium, the dynamics of this automaton is far from simple. In fact, for  $d \geq 3$  and other topologies, it has been shown to be P-complete.<sup>24,25</sup> It is not difficult to construct wires and some logical gates by using tokens: in Figure 1.19 we exhibit this constructions in dimension two.<sup>24</sup>

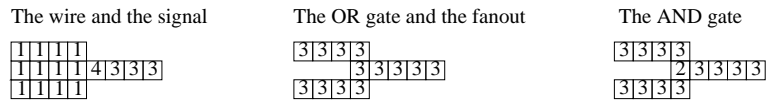


Fig. 1.19. The logical gates for the sandpile model.

However, the construction of a CROSSOVER in two dimensions is not possible and this can be proved;<sup>26</sup> here we sketch the argument. A CROSSOVER can be viewed as a stable configuration in a finite portion of the 2-dimensional lattice. The circuit activity begins when some tokens are added to the input cells provoking avalanches. In order to cross avalanches, the CROSSOVER must be susceptible to develop a West-to-East and a South-to-North avalanche. The difficulty is that the avalanches intersect each other, because of the planarity of the grid, and the monotonicity of the rule makes one avalanche follow the course of the other one and vice versa. Figure 1.20 shows a configuration that could be a CROSSOVER. It can produce both West-to-East and a South-to-North avalanches. But when one of these avalanches is produced, tokens arrives both to the East and North sides.

The same result is obtained when we consider Moore's neighborhood (*i.e.*, the 8 neighbors of a site). But if we relax the size of the neighborhood further by considering radius 2, one can obtain a P-complete two dimensional model,<sup>26</sup> whose logic gates as well as CROSSOVER are shown in

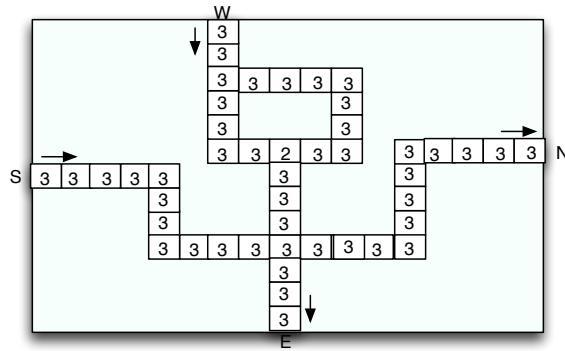


Fig. 1.20. A failed CROSSOVER. If an avalanche comes in by the West or by the South, it exits both by the North and the East.

Figure 1.21.

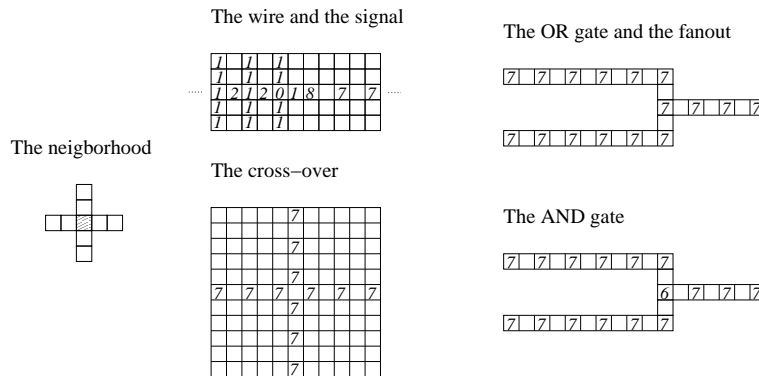


Fig. 1.21. Logical gates constructed for a Sandpile with von Neumann neighborhood of radius two. In this model, the critical threshold is 8.

### 1.7. Conclusions

The construction of boolean circuits in 2-CA is not always an easy task, but it remains the best known path to proving Turing Universality. Turing Universality is an important property, since it means being able to perform any algorithmic calculation, and has strong consequences like the existence

of undecidable questions (in particular, that of the halting problem, which can be translated for a T.U. CA as the undecidability of knowing whether a certain local configuration will ever appear on the grid). But even when Turing Universality is not reached, *Circuit Universality* is by itself an interesting feature. If all the needed devices are available and can be properly combined, then configurations can be built to compute any given function  $F : \{0, 1\}^N \rightarrow \{0, 1\}^M$ ; moreover, the  $P \neq NC$  conjecture would put the dynamics of the CA in the “inherently sequential” category, making easy prediction hopeless. These two consequences of circuit construction are enough to justify it as a relevant certificate of complexity.

The set of models shows both the similarities and the diversity we can encounter when we try to follow Banks’s steps toward universality. The shape and cardinality of the neighborhood can be a problem, sometimes solvable (like the case of the hexagonal grid, where it prevented an easy migration of some of Banks’s constructions) and sometimes insurmountable (witness, the impossibility of crossovers in the standard sandpile). In some cases moving configurations are readily available (Life’s gliders, billiard balls), but often a signal must be coded into a wire. The logic gates that are easier to build vary greatly between different automata too; and sometimes it is not in the logic gates, but in the complementary devices, where the most difficult part of the construction is found: duplicating, producing or even directing signals can be non-trivial, and even garbage disposal can become an issue.

Banks’s approach or slight variants of it have even worked well in quite different settings, like Langton’s ant,<sup>27</sup> where changes on the lattice are made by the action of an agent (the ant) while the rest of the configuration remains static. Circuit and Turing universality were proved for this system by using gates and devices very much like those described here, the main differences being that there the configurations must be defined in such a way that the ant visits each gate to make it work, while logical values are “read” by the ant by affecting its trajectory, and “written” through the trail it leaves behind.<sup>28</sup>

Are there precise features shared by all these automata, which allow them to be Circuit Universal? A set of sufficient conditions seems unlikely, but perhaps some necessary conditions could be found which the local rule of a 2-CA must satisfy if arbitrary boolean circuits are to be constructed. This is an intriguing open question, since proving non-universality is even more difficult, for non-trivial CA, than proving the opposite.

## Acknowledgments

We want to thank particularly to Andrés Moreira for careful reading and precious comments.

## References

1. R. Greenlaw, H. Hoover, and W. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory*. (Oxford University Press, 1995).
2. R. E. Ladner, The circuit value problem is log space complete for P, *SIGACT News*. **7**(1), 18–20, (1975).
3. E. R. Banks. *Information Processing and Transmission in cellular automata*. PhD thesis, M.I.T., Cambridge, Mass., U.S.A., (1971).
4. B. Durand and Z. Róka, *The game of life: universality revisited*, In eds. M. Delorme and J. Mazoyer, *Cellular Automata: a parallel model*, pp. 51–76. Kluwer Academic Pub., (1999).
5. N. Ollinger. *Automates cellulaires : structures*. PhD thesis, École Normale Supérieure de Lyon, France, (2002).
6. J.-C. Delvenne, P. Kůrka, and V. D. Blondel, Computational universality in symbolic dynamical systems, *Fundamenta Informaticae*. **74**:4, 463–490, (2006).
7. M. Cook, Universality in elementary cellular automata, *Complex Systems*. **15**(1), 1–40, (2004).
8. S. Wolfram, *A new kind of science*. (Wolfram Media Inc., 2002).
9. L. M. Goldschlager, The monotone and planar circuit value problems are log space complete for P, *SIGACT News*. **9**(2), 25–29, (1977).
10. A. K. Dewdney, Computer recreations: The cellular automata programs that create Wireworld, Rugworld and other diversions, *Scientific American*. **262**(1), 146–149 (January, 1990).
11. M. Owen, (2004). <http://www.quinapalus.com/wi-index.html>.
12. A. Gajardo and E. Goles, Universal cellular automaton over a hexagonal tiling with 3 states, *International Journal of Algebra and Computation*. **11** (3), 335–354, (2001).
13. E. Berlekamp, J. H. Conway, and R. Guy, *Winning Ways for Your Mathematical Plays*. vol. 2, (Academic Press., 1982).
14. D. Griffeath and C. Moore, Life without death is P-complete, *Complex Systems*. **10**, 437–447, (1996).
15. E. Fredkin and T. Toffoli, Conservative logic, *Int. J. of Theoret. Phys.* **21** (3/4), 219–253, (1982).
16. K. Morita, A simple construction method of a reversible finite automaton out of Fredkin gates, and its related problem, *The Trans. of the IEICE*. **E73** (6), 978–984, (1990).
17. N. Margolus. *Physics and Computation*. PhD thesis, M. I. T., Cambridge, Mass., U.S.A., (1987).
18. J. Hardy, O. de Pazzis, and Y. Pomeau, Molecular dynamics of a classical

- lattice gas: transport properties and time correlation functions, *Phys. Rev. A*. **13**, 1949–1960, (1976).
19. K. Morita and S. Ueno, Computation-universal models of two-dimensional 16-state reversible cellular automata, *IEICE Trans. Inf. and Syst.* **E75-D**(1), 141–147, (1992).
  20. K. Morita, M. Margenstern, and K. Imai, Universality of reversible hexagonal cellular automata, *RAIRO Theoretical Informatics and Applications.* **33**, 535–550, (1999).
  21. K. Imai and K. Morita, A computation-universal two-dimensional 8-state triangular reversible cellular automaton, *Theoret. Comput. Sci.* **231**, 181–191, (2000).
  22. C. Moore and M. G. Nordahl. Predicting lattice gases is P-complete. Technical Report 034, Santa Fe Institute, Santa Fe, New Mexico (apr, 1997).
  23. P. Bak, C. Tang, and K. Wiesenfeld, Self-organized criticality: An explanation of  $1/f$  noise, *Phys. Rev. Lett.* **59**(4), 381–384, (1987).
  24. C. Moore and M. Nilsson, The computational complexity of Sandpiles, *J. of Stat. Phys.* **96**, 205–224, (1999).
  25. E. Goles and M. Margenstern, Sand pile as a universal computer, *Int. J. of Modern Physics C.* **7**(2), 113–122, (1996).
  26. A. Gajardo and E. Goles, Crossing information in two dimensional Sandpiles, *Theor. Comput. Sci.* **369**(1-3), 463–469, (2006).
  27. C. G. Langton, Studying artificial life with cellular automata, *Physica D.* **22**, 120–149, (1986).
  28. A. Gajardo, A. Moreira, and E. Goles, Complexity of Langton’s ant, *Discrete Applied Mathematics.* **117**, 41–50, (2002).