# Simulating 3D Cellular Automata with 2D Cellular Automata

Victor Poupet

LIP (UMR CNRS, ENS Lyon, INRIA, Univ. Claude Bernard Lyon 1),
École Normale Supérieure de Lyon,
46 allée d'Italie 69364 LYON cedex 07 FRANCE

**Abstract.** The purpose of this article is to describe a way to simulate any 3-dimensional cellular automaton with a 2-dimensional cellular automaton. We will present the problems that arise when changing the dimension, propose solutions and discuss the main properties of the obtained simulation.

## 1 Introduction

Cellular automata are a widely studied and massively parallel computing model. They are composed of *cells*, arranged on $\mathbb{Z}^n$ (where $n$ is the *dimension* of the automaton). The cells can be in different *states*, the set of states being finite and common to all cells. They *evolve* synchronously in a deterministic way at discrete times. All cells evolve according to the same local rule, and their new state depends only on their neighbors' states. From this local behavior, we can define a global evolution, from any configuration of the automaton into another, by having each cell change its state according to the local rule.

Despite their apparent simplicity cellular automata can have very complex behaviors and are commonly used in order to modelize various phenomena. The underlying graph is usually $\mathbb{Z}^2$ or $\mathbb{Z}^3$ (depending on the space of the modelized reality). Computer networks however are often massively parallel computers working on $\mathbb{Z}$ or $\mathbb{Z}^2$ in which $\mathbb{Z}^3$ cannot be embeded easily. Various attempts have been made to simulate a $\mathbb{Z}^3$ space on a $\mathbb{Z}^2$ network: for example using hexagonal networks (see [1]) but in this case a degree of liberty is lost [2]. These simulations also lead to a problem of representation of the usual $\mathbb{Z}^3$ shapes (lines, spheres, planes, cones etc.) in $\mathbb{Z}^2$ without loss of information.

The subject matter of this article is to show how it is possible, given a 3-dimensional cellular automaton (3DCA) $\mathcal{A}_3$, to construct a 2-dimensional cellular automaton (2DCA) $\mathcal{A}_2$ that will *mimic* its behavior, in a sense that we will define. To do so, we want to associate to each cell of $\mathcal{A}_3$ a unique cell of $\mathcal{A}_2$ with a one-to-one function $t$, so that $t(c)$ will simulate the evolution of $c$. Considering the relative sizes of the spheres of radius $k$ in $\mathbb{Z}^3$ and $\mathbb{Z}^2$, it is obvious that there will be cells $c_1$ and $c_2$ that are very close in $\mathbb{Z}^3$ and such that $t(c_1)$ and $t(c_2)$ are arbitrarily far in $\mathbb{Z}^2$ (see [3]). For this reason, it won't be possible to execute the simulation in linear time, but we will show that it is possible to have $\mathcal{A}_2$ compute $k$ generations of $\mathcal{A}_3$ in polynomial time.

In this article, for matters of simplicity, we will consider that $\mathcal{A}_3$ works on the 3D Von Neumann neighborhood $\mathcal{V}_3 = \{\overrightarrow{0}, \pm\overrightarrow{x}, \pm\overrightarrow{y}, \pm\overrightarrow{z}\}$ and that $\mathcal{A}_2$ works on the Moore neighborhood $\mathcal{V}_2$ (9 neighbors including itself). There is no real loss of generality since simulating different neighborhoods in linear time is a well known technique (see [4]).

Now if we assume that all cells in $t(\mathbb{Z}^3)$ have been marked on $\mathcal{A}_2$ by a particular state and also that for each $c \in \mathbb{Z}^3$, $t(c)$ "knows" the state of $c$ in $\mathcal{A}_3$, then $t(c)$ will send signals indicating this state in the 6 directions *up*, *up-right*, *right*, *down*, *down-left* and *left*.

If we can make $t$ so that the first cells in $t(\mathbb{Z}^3)$ that these signals meet are precisely the images of $c$'s neighbors by $t$ (that we will call the $t$-neighbors of $t(c)$), $t(c+\overrightarrow{x})$ receiving the signal going *right*, $t(c+\overrightarrow{y})$ the one going *up*, $t(c+\overrightarrow{z})$ the one going *up-right* and symmetrically for their opposites, then each cell in $t(\mathbb{Z}^3)$ gets the state of all its $t$-neighbors and can apply the rule of $\mathcal{A}_3$, and send its state again...

## 2 The Projection $t$ from $\mathbb{Z}^3$ to $\mathbb{Z}^2$

In order to be able to transmit the current state of a cell of $t(\mathbb{Z}^3)$ to all its $t$-neighbors as explained previously, it is necessary that the function $t$ is such that for all $(a,b,c) \in \mathbb{Z}^3$, all the points $t(a+k,b,c)_{k\in\mathbb{Z}}$ are on the same horizontal line and ordered according to $k$. Moreover, if $(b,c) \neq (b',c')$ then $t(a,b,c)$ and $t(a,b',c')$ must be on different horizontal lines. Similarly, all the points $t(a,b+k,c)_{k\in\mathbb{Z}}$ must be on the same column (one for each couple $(a,c)$), and $t(a,b,c+k)_{k\in\mathbb{Z}}$ on the same diagonal (one for each $(a,b)$).

We prove easily that these requirements are equivalent to the existence of $f$, $g$, and $h$ three increasing functions from $\mathbb{Z}$ into $\mathbb{Z}$ such that

$$\forall (a,b,c) \in \mathbb{Z}^3, \qquad t(a,b,c) = (f(a)+h(c), g(b)+h(c))$$

$$\left.\begin{array}{l}(x,z) \mapsto f(x)+h(z)\\ (y,z) \mapsto g(y)+h(z)\\ (x,y) \mapsto g(y)-f(x)\end{array}\right\} \quad \text{are one-to-one.} \tag{1}$$

There are many functions that verify the equations (1), for example

$$f,g,h : \mathbb{Z} \to \mathbb{Z} \quad \begin{cases} f(k) = \mathrm{sgn}(k).2^{3|k|} \\ g(k) = \mathrm{sgn}(k).2^{3|k|+1} \\ h(k) = \mathrm{sgn}(k).2^{3|k|+2} \end{cases} \text{(where sgn}(x) \text{ is the sign of } x)$$

However, we will focus on a solution where the functions grow polynomially. We have the following theorem:

**Theorem 1.** *There exist three increasing functions $f$, $g$ and $h$ in $\mathbb{Z} \to \mathbb{Z}$, bounded by polynomials of degree 3 and satisfying the equations (1).*

*Proof.* Consider the three functions $f$, $g$ and $h$ defined by induction as follows:

- All three functions are odd ($f(0) = 0$ and $\forall x, f(-x) = -f(x)$).
- For all $n \in \mathbb{N}$, $f(n+1)$ is defined as the smallest positive integer not in $\{f(i) + g(j) + g(k), \quad f(i) + h(j) + h(k) \quad \big| \quad |i|, |j|, |k| \leq n\}$
- For all $n \in \mathbb{N}$, $g(n+1)$ is defined as the smallest positive integer not in $\{g(i) + h(j) + h(k), g(i) + f(j') + f(k') \quad \big| \quad |i|, |j|, |k| \leq n, |j'|, |k'| \leq n+1\}$
- For all $n \in \mathbb{N}$, $h(n+1)$ is defined as the smallest positive integer not in $\{h(i) + f(j) + f(k), \quad h(i) + g(j) + g(k) \quad \big| \quad |i|, |j|, |k| \leq n+1\}$

We can easily see that these functions verify the equations (1). Moreover,

$$\forall n \in \mathbb{N}, \quad \begin{cases} \text{Card}(\{f(i) + g(j) + g(k) | -n \leq i, j, k \leq n\}) \leq (2n+1)^3 \\ \text{Card}(\{f(i) + h(j) + h(k) | -n \leq i, j, k \leq n\}) \leq (2n+1)^3 \end{cases}$$

Since both sets contain 0, their union is of cardinal lower than $2(2n+1)^3$. From the definitions of $f$, $g$ and $h$, we get

$$\forall n \in \mathbb{N}, \quad \begin{cases} f(n+1) \leq 2(2n+1)^3 & \leq 2(2n+3)^3 = O(n^3) \\ g(n+1) \leq 2(2n+1)(2n+3)^2 & \leq 2(2n+3)^3 = O(n^3) \\ h(n+1) \leq 2(2n+1)(2n+3)^2 & \leq 2(2n+3)^3 = O(n^3) \end{cases} \tag{2}$$

□

From now on, we will consider that the functions $f$, $g$ and $h$ are the ones defined in the proof of the theorem 1, and the function $t$ is the one defined by (1).

The table 1 gives the first 10 values of the $f$, $g$ and $h$ functions. We can already see that these functions are very irregular. Although we have a polynomial upper bound (2), the functions don't seem to increase regularly: $g(5)$ and $g(6)$ are extremely close for example. Moreover, from the definitions, it seems that $f$ will be inferior to $g$ and $h$ because its value is chosen before the other two. However, we have $g(7) >> f(7)$ and $h(8) >> f(8)$. For these reasons, it seems very hard to give a good lower bound of these functions.

## 3 Construction of $t(\mathbb{Z}^3)$ by a 2DCA

In this section, we want to show that $t(\mathbb{Z}^3)$ can be constructed by a cellular automaton in polynomial time, meaning that there exists a polynomial $P \in \mathbb{Z}[X]$ such that for all $p \in \mathbb{Z}^3$, $t(p)$ is constructed at time at most $P(\|p\|)$. We have the following theorem:

**Theorem 2.** *There exists a 2DCA working on the Moore neighborhood that, starting from a configuration where all cells are in the quiescent state except the origin, will* mark *by a particular state all cells that are in $t(\mathbb{Z}^3)$ in polynomial time.*

**Table 1.** The first values of the functions $f$, $g$ and $h$

| $n$ | $f(n)$ | $g(n)$ | $h(n)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 3 | 4 |
| 2 | 10 | 15 | 17 |
| 3 | 26 | 38 | 46 |
| 4 | 57 | 75 | 65 |
| 5 | 84 | 116 | 128 |
| 6 | 143 | 119 | 176 |
| 7 | 152 | 223 | 193 |
| 8 | 270 | 276 | 340 |
| 9 | 327 | 380 | 386 |
| 10 | 510 | 553 | 579 |

### 3.1 Computations on the Axis

The first step of the computation is to mark all the points on the horizontal axis that correspond to the image of $\mathbb{N}$ by $f$, $g$ and $h$. Given the recursive definitions of the functions, the automaton will successively mark $f(1)$, $g(1)$, $h(1)$, $f(2)$ etc.

The general idea is to mark, after $f(k)$ is constructed, all the points of the form $f(k)\pm g(i)\pm g(j)$, $f(k)\pm h(i)\pm h(j)$, $g(i)\pm f(j)\pm f(k)$ and $h(i)\pm f(j)\pm f(k)$ for all $i, j < k$, and symmetrically after marking $g(k)$ or $h(k)$. If we inductively assume that all the necessary markings have been done, then after marking $f(k+1)$ and all the corresponding points, a signal can move to the right from $g(k)$ until it reaches the first cell that is not "forbidden" by the definition of $g$ and mark it as being $g(k+1)$.

Also, from the start of the construction, the horizontal axis ($y = 0$), the vertical axis ($x = 0$) and the "diagonal axis" ($x = y$) will be marked by signals going at maximum speed.

We will only focus on the construction of the functions on the positive half axes, but the symmetric construction will be made on the negative half axes.

**Addition and Substraction on the Axes.** Let us consider two integers $a$ and $b$ such that $0 < a \leq b$. We assume that both cells $(a, 0)$ and $(b, 0)$ have been marked on the axis at times $\tau(a)$ and $\tau(b)$ respectively. Then, using constructions as illustrated in figures 1 and 2, it is possible to mark the points $(a + b, 0)$ and $(b - a, 0)$ at times

$$\begin{array}{l} \tau(a+b) = \max(\tau(a) + a + b, \tau(b) + 2a) \quad \text{and} \\ \tau(a-b) = \max(\tau(a) + b, \tau(b) + b) \ . \end{array} \tag{3}$$

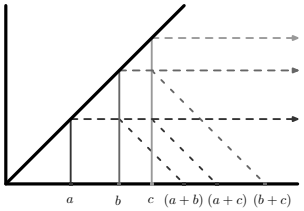


**Fig. 1.** Additions on a 2DCA

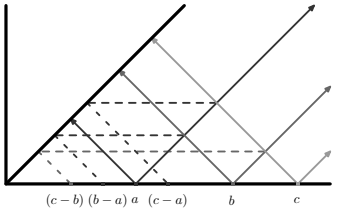


**Fig. 2.** Substractions on a 2DCA

### 3.2 Computing the Functions on the $X$-Axis

In the following section, we will identify the $x$-axis of $\mathbb{Z}^2$ with $\mathbb{Z}$. In other words, we will refer to the cell $(n, 0)$ simply as $n$.

To construct the sets $f(\mathbb{N})$, $g(\mathbb{N})$ and $h(\mathbb{N})$ we will use many different states. Let's assume that at time $\tau(n)$ all the following cell sets are marked on the $x$-axis (some cells might be in several sets in which case it is marked with a "combination" of states):

$$\begin{array}{l} F = \{f(k), k \leq n\}, \\ 2F = \{f(k) \pm f(k') | k, k' \leq n\} = F \pm F \text{ and} \\ \overline{F} = (\pm F \pm 2G \cup \pm F \pm 2H) \cap \mathbb{N} \ . \end{array}$$

and the corresponding sets after permutation of $f$, $g$ and $h$.

Also, we assume that $f(n)$, $g(n)$ and $h(n)$ are in particular states that indicate they are the greatest known points of $f(\mathbb{N})$, $g(\mathbb{N})$ and $h(\mathbb{N})$ and that every point marked on the $x$-axis (as being part of any of the previous sets) has sent signals diagonally to his right and his left in order to be able to perform additions and substractions as illustrated in the previous paragraph.

From here, constructing $f(n+1)$ is simply a matter of moving a signal from $f(n)$ to the right until it reaches a cell that is not marked as being in $\overline{F}$. $f(n)$ "forgets" that it is the last value of $F$ when the signal starts searching for $f(n+1)$ and when $f(n+1)$ is marked, it "knows" that it is the last known value of $F$. $f(n+1)$ is also marked as being part of the sets $F$, $2F$, $\overline{F}$, $\overline{G}$ and $\overline{H}$. New diagonal signals are sent to the right and the left, that will interact with the existing ones (sent by previously marked points) to mark the cells of the following sets:

$$\begin{array}{l} f(n+1) \pm F \subseteq 2F, \\ 2f(n+1) \in 2F, \\ \pm f(n+1) \pm 2G \text{ and} \\ \pm f(n+1) \pm 2H \subseteq \overline{F} \ . \end{array}$$

and then the points in $2F + G \subseteq \overline{G}$ and $2F + H \subseteq \overline{H}$ since we have changed $2F$.

In order for the point $g(n+1)$ to be marked by the same method without error, we have to make sure that the set $\overline{G}$ has been completely marked. It is easy to see that the last point of this set to be constructed will be $2f(n+1) + 2g(n)$.

Using the inequations from (3), we see that it'll be constructed at most at time

$$\begin{aligned}&\big[\tau(f(n+1)) + 2f(n+1)\big] + 2\big[f(n+1) + 2g(n)\big]\\ &= \tau(f(n+1)) + 4(f(n+1) + g(n))\ .\end{aligned}$$

where $\tau(f(n+1))$ is the time when $(f(n+1), 0)$ is marked as being in $F$.

Therefore, we have to wait $4f(n+1) + 4g(n)$ generations before we start looking for $g(n+1)$. To do so, the cell $f(n+1)$ will send two signals as soon as it is marked. The first one will move to the left at speed $1/4$ while the second one will move to the right at speed 1. Their evolution will only be affected by the cells $f(n+1)$, $g(n)$ and the origin $(0, 0)$. We are in one of these two possibilities (both cases are illustrated by figures 3 and 4):

- $0 < g(n) < f(n+1)$, in this case the right signal won't encounter any important cell and thus will continue to the right forever. On the other hand, the left signal will reach $g(n)$ and remember it (by changing its state), and continue to the origin still at speed $1/4$. After reaching the origin, it will turn around and return to $g(n)$ at speed $1/4$.
  When the right signal arrives at $g(n)$ for the second time, it has travelled during $4f(n+1) + 4g(n)$ generations.
- $0 < f(n+1) < g(n)$, in this case, the left signal will go directly to the origin without ever seeing $g(n)$ and will disappear. The right signal however will reach $g(n)$, turn around and go back to $f(n+1)$ at speed 1, then continue to the origin at speed $1/4$ where it turns around and goes to $f(n+1)$ for the second time at speed $1/4$, and finally goes to $g(n)$ at speed $1/2$.
  When it reaches $g(n)$ for the second time, the right signal has travelled during $4f(n+1) + 4g(n)$ generations.

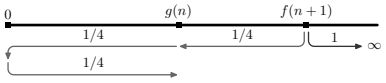


**Fig. 3.** $0 < g(n) < f(n+1)$

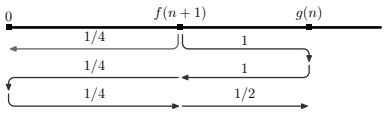


**Fig. 4.** $0 < f(n+1) < g(n)$

In both cases, we see that one of the signals reaches $g(n)$ exactly when the last point of $\overline{G}$ is constructed on the axis, and so $g(n)$ can send a signal to the right to mark $g(n+1)$. After marking $g(n+1)$, two signals are sent to wait exactly $4g(n+1) + 4h(n)$ generations while the set $\overline{H}$ is being updated. Then $h(n+1)$ is constructed, and after waiting long enough, a signal reaches $f(n+1)$ when $\overline{F}$ is up to date. At this point we are in the same conditions that the ones we assumed to start constructing $f(n+1)$ and so the inductive construction of $f(\mathbb{N})$, $g(\mathbb{N})$ and $h(\mathbb{N})$ can continue.

We have the following inequalities:

$$\begin{aligned}
&\tau(g(n+1)) \geq \tau(f(n+1)) + 4f(n+1) + 4g(n) \\
&\tau(h(n+1)) \geq \tau(g(n+1)) + 4g(n+1) + 4h(n)) \\
&\tau(f(n+2)) \geq \tau(h(n+1)) + 4h(n+1) + 4f(n+1))
\end{aligned} \tag{4}$$

**Polynomial Time Bounds.** We will now give a rough upper bound of the time needed to construct the functions $f$, $g$ and $h$ on the $x$-axis. We assume that $\tau(f(0)) = \tau(g(0)) = \tau(h(0)) = 0$ since the state of the origin is given in the initial configuration. After marking $f(n+1)$, we wait exactly $4f(n+1) + 4g(n)$ generations before the signal starts propagating to the right from $g(n)$ and $g(n+1) - g(n)$ before it reaches its destination.

Therefore we have, using the inequalities from (2), for all $n \in \mathbb{N}$

$$\tau(g(n+1)) = \tau(f(n+1)) + O(n^3)$$

And using the corresponding result on $f$ and $h$, for all $n \in \mathbb{N}$

$$\tau(g(n+1)) = \tau(g(n)) + O(n^3) = \sum_{k=0}^{n} O(k^3) = O(n^4) \tag{5}$$

We have the same polynomial bound on $\tau(f(n))$ and $\tau(h(n))$.

**Construction on the Other Axes.** When the cell $(g(n), 0)$ is marked as being part of $G$, a signal is sent in the direction $(-1, 1)$ and it reaches the $y$-axis on $(0, g(n))$ after $g(n)$ generations and marks it. Similarly, when $h(n)$ is marked, a signal is sent in the direction $(0, 1)$ that reaches the diagonal axis $(x = y)$ on $(h(n), h(n))$ after $h(n)$ generations.

### 3.3 Computation of the Rest of $t(\mathbb{Z}^3)$

We will consider here the norm 1 on $\mathbb{Z}^3$ ($\|(a, b, c)\| = |a| + |b| + |c|$). The sphere of radius $k$ for this norm will be denoted as $S_k$.

Let $p = (x, y, z) \in \mathbb{Z}^3$, with $\|p\| = k$. We will say that $p' = (x', y', z')$ is a *superior neighbor* of $p$ (and that $p$ is an *inferior neighbor* of $p'$) if $p' \in S_{k+1}$ and $\|p' - p\| = 1$. In $\mathbb{Z}^2$ we will say that $t(p')$ is a *superior $t$-neighbor* of $t(p)$. The number and the position (relative) of the superior neighbors of $p$ only depend on the signs of $p$'s coordinates. Also, every point that is not on one of the axes (in $\mathbb{Z}^3$) has at least two inferior neighbors (one for each non-zero coordinate).

**Inductive Construction.** We will see here how it is possible to inductively construct the images by $t$ of all the spheres $S_k$. Since it is not possible to give each cell in $t(\mathbb{Z}^3)$ the number of the sphere it is in (this would require an infinite number of states) we will consider two cell sets $S_{+0}$ and $S_{+1}$ that will represent, for each step $k$ of the induction, the current sphere ($S_k$) and the next one ($S_{k+1}$). During step $k$ we will construct $S_{k+2}$.

For $k_0 \in \mathbb{N}$, let's assume we are in a configuration such that:

– For all $p \in S_{k_0}$, the cell $t(p)$ is in $S_{+0}$ and knows the sign of each of $p$'s coordinates (and so knows the direction in which each of its superior $t$-neighbors is). Reciprocally, every cell in $S_{+0}$ is the image of a point of $S_{k_0}$.
– For all $p \in S_{k_0+1}$, the cell $t(p)$ is in $S_{+1}$ and knows the sign of each of $p$'s coordinates (it knows where its inferior $t$-neighbors are). Also, every cell in $S_{+1}$ is the image of a point of $S_{k_0+1}$.

Such a situtation is illustrated on the figure 5 for $k_0 = 1$.

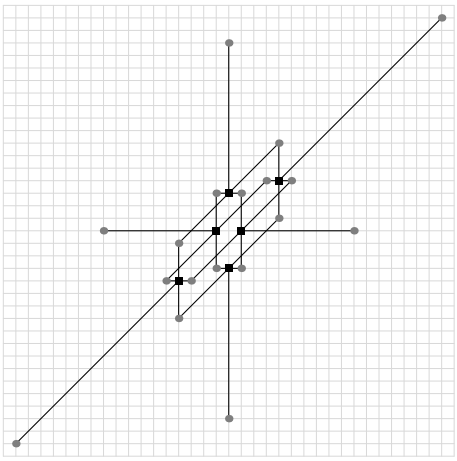

**Fig. 5.** The spheres $S_1$ *(black squares)* and $S_2$ *(grey circles)*

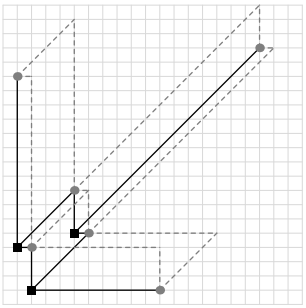

**Fig. 6.** The signals $s_1$ *(black)* and $s_2$ *(dashed grey)* during step 1

In this situation, a signal $s_0$ is sent from the origin (we will explain later what causes this signal to appear) in all directions and spreads at maximum speed forming a square around the origin. When a cell in $S_{+0}$ receives this signal, it sends a signal $s_1$ to each of its superior $t$-neighbors and, after doing so, is no longer in $S_{+0}$. When a cell in $S_{+1}$ receives a signal $s_1$, the signal disappears and the cell sends a signal $s_2$ to each of its superior $t$-neighbors except the one that is the direction of the received signal (this is illustrated in figure 6 for the upper right quarter of $\mathbb{Z}^2$). The upper $t$-neighbors of these cells are not yet constructed but the cells know in which direction they are because they know the coordinates of their corresponding point in $\mathbb{Z}^3$. When a cell in $S_{+1}$ has received a signal $s_1$ from all its inferior $t$-neighbors it becomes a cell of $S_{+0}$ and is no longer in $S_{+1}$. Since the signal $s_0$ spreads at maximum speed we are sure that it cannot interact with these cells (but it will during the next step of course).

When two $s_2$ signals meet on a cell, they both disappear and the cell is marked as being in $S_{+1}$ (again, this cell won't do anything before the next step since it won't receive any $s_1$ signal). Also, the direction of the $s_2$ signals that arrive to a cell indicate the position of its inferior $t$-neighbors so when all signals

have arrived, the cell knows where all its superior $t$-neighbors are. Note that a given cell can (and most of the time will) receive more than one couple of $s_2$ signals. Each time, the signals disappear, and the cell gets more information from the inferior $t$-neighbors.

This construction marks all cells in $t(S_{k_0+2})$ that have at least two inferior $t$-neighbors. Only the ones on the axes are left, and they are constructed independently as explained earlier.

**Correctness of the Construction.** The correctness of this construction is based on two observations. First, there cannot be any conflict between the $s_1$ and $s_2$ signals because between a cell of $t(\mathbb{Z}^3)$ and one of its superior $t$-neighbors there is no other cell of $t(\mathbb{Z}^3)$. Second, if we consider a point $p \in S_{k_0+2}$ having at least two inferior neighbors $p_1$ and $p_2$, then these two points share a common inferior $t$-neighbor $p'$ because $p_1$ and $p_2$ can be obtained from $p$ by reducing one of the coordinates (a different one for each), and so $p'$ can be obtained by reducing both coordinates from $p$. Moreover, the basic properties of $t$ ensure that the points $t(p)$, $t(p_1)$, $t(p_2)$ and $t(p')$ form a parallelogram. So the paths $p' \to p_1 \to p$ and $p' \to p_2 \to p$ have same length and so the two $s_2$ signals will meet on $p$ (see figure 6).

Every cell in $t(S_{k_0+1})$ will therefore receive a couple of coinciding $s_2$ signals for each couple of its inferior neighbors, and so will be able to deduce the signs of all coordinates of the point in $\mathbb{Z}^3$ it is the image of.

**Synchronisation.** The previously explained construction relies on a signal $s_0$ that will initiate it. This signal must appear after all points of $S_{k_0}$ and $S_{k_0+1}$ are marked as being in $S_{+0}$ and $S_{+1}$ respectively. Since the construction is inductive, it is obvious that $S_{k_0+1}$ will be marked after $S_{k_0}$. Therefore, it is sufficient to ensure that the signal $s_0$ (corresponding to step $k_0$) appears after $S_{k_0+1}$ is fully marked.

To do so, let's create a signal from $(h(k_0+1), 0)$ when the cell is marked as being in $h(\mathbb{N})$ (at time $\tau(h(n))$) that will move left at speed 1, and will trigger the $s_0$ signal when reaching the origin.

Let's prove the construction by induction. For some $k_0 \in \mathbb{N}$, let's assume that all points of $S_{k_0+1}$ are marked at time $\tau_s(k_0) = \tau(h(k_0+1)) + h(k_0+1)$ (when the signal $s_0$ from step $k_0$ appears). Then $s_0$ reaches all points in $S_{k_0}$ by time at most

$$\tau_s(k_0) + \max(f(k_0), g(k_0)) + h(k_0)$$

and the propagation of signals $s_1$ and $s_2$ lasts at most

$$2\max(f(k_0+1) - f(k_0), g(k_0+1) - g(k_0), h(k_0+1) - h(k_0))$$

In the end, all points of $S_{k_0+1}$ that are marked by these signals are marked at time at most

$$\tau(h(k_0+1)) + h(k_0+1) + \max\begin{cases} f(k_0) + h(k_0) \\ g(k_0) + h(k_0) \end{cases} + 2\max\begin{cases} f(k_0+1) - f(k_0) \\ g(k_0+1) - g(k_0) \\ h(k_0+1) - h(k_0) \end{cases}$$

It is easy to see that this time is lower than

$$\tau(h(k_0+1)) + h(k_0+1) + 2f(k_0+1) + 2g(k_0+1) + 2h(k_0+1)$$

And if we use the inequalities from (4) we get

$$\begin{aligned}\tau_s(k_0+1) &= t(h(k_0+2)) + h(k_0+2)\\ &\geq \tau(h(k_0+1)) + 4h(k_0+1) + 4f(k_0+2) + 4g(k_0+2) + h(k_0+2)\\ &\geq \tau(h(k_0+1)) + h(k_0+1) + 2f(k_0+1) + 2g(k_0+1) + 2h(k_0+1)\end{aligned}$$

Also, the only points in $t(S_{k_0+2})$ that are not constructed by the signals $s_2$ are the ones on the axes. There are six of them: $\pm(f(k_0+2),0)$, $\pm(0,g(k_0+2))$ and $\pm(h(k_0+2),h(k_0+2))$, and they are marked at times $\tau(f(k_0+2))$, $\tau(g(k_0+2))+g(k_0+2)$ and $\tau(h(k_0+2))+h(k_0+2)$ respectively, all lower than or equal to $\tau_s(k_0+1)$.

Therefore, when the signal $s_0$ from step $k_0+1$ is triggered all points of $S_{k_0+1}$ and $S_{k_0+2}$ are marked and the construction can be done correctly.

### 3.4 End of the Construction

The last thing to do now is to "activate" the cells in $t(\mathbb{Z}^3)$ when all of their $t$-neighbors have been constructed (we will need this information later when simulating $\mathcal{A}_3$). To do so, since every point of $t(\mathbb{Z}^3)$ knows the direction where all its inferior $t$-neighbors are when it is constructed, it will send a special "activation" signal to them as soon as it is marked. When a given cell has received an "activation" signal from all its superior $t$-neighbors it goes into a new "activated" state.

We have therefore seen in this section how it is possible to construct, using a 2DCA working on Moore's neighborhood starting on the configuration where all cells are in a quiescent state except the origin, the image of $\mathbb{Z}^3$ by $t$. This construction is done in polynomial time, meaning that the image of the sphere $S_k$, which is of radius $O(k^3)$ as seen in (2) is constructed in time $O(k^4)$ (according to (5) and the arguments of induction).

This ends the proof of theorem 2.

## 4 Description of $\mathcal{A}_2$

The final 2DCA $\mathcal{A}_2$ is a superposition of two 2DCA working in parallel. The *lower layer* is the one that will mark $t(\mathbb{Z}^3)$ during the simulation as explained previously. We will represent the evolution of this construction by colors. A cell that has been marked as part of $t(\mathbb{Z}^3)$ but has not yet been *activated* (its $t$-neighbors are still being constructed) will be represented as *grey*. A cell marked and activated will be either *black* or *red* (we will see later why we need two colors) and a cell that hasn't been marked is *white*. We will say that a cell is *colored* if it is either *grey*, *black* or *red* (as opposed to *white*). The *upper layer* is the one that will really do the simulation.

Colored cells all correspond to a cell of $\mathcal{A}_3$. Their state will be a product of information. What we will call their central state is the state of their corresponding cell that they are currently simulating. A *red* cell will send signals containing its central state in the directions of its $t$-neighbors. The next generation, it becomes *black* so that the central state is sent only once. These signals are transmitted by *white* cells. When a *black* cell receives one of these signals, the signal disappears and the cell memorizes the state of the corresponding $t$-neighbor (which neighbor depends on the direction the signal came from). When it knows all the central states of its $t$-neighbors, the *black* cell applies the transition function of $\mathcal{A}_3$, changes its central state and becomes *red* (to transmit the new central state)...

Given a configuration $\mathcal{C}_3 \in \mathcal{Q}_3^{\mathbb{Z}^3}$, the initial configuration of $\mathcal{A}_2$ is as follows. The lower layer is initialized on the origin only (all other cells are in the blank state) to construct $t(\mathbb{Z}^3)$ as we have seen. If $\mathcal{C}_3$ is finite (only a finite number of cells are not in the blank state $q_0$), for all $c \in \mathbb{Z}^3$ such that $\mathcal{C}_3(c) \neq q_0$, $t(c)$ receives $\mathcal{C}_3(c)$ as its central state. All other cells are left blank, so the initial configuration of $\mathcal{A}_2$ is finite too. If $\mathcal{C}_3$ is infinite, all cells in $t(\mathbb{Z}^3)$ receive the state of their preimage.

As the construction of $t(\mathbb{Z}^3)$ by the lower layer goes on, cells become *grey* (marked but not activated) and then they are activated and become *red*. A new *red* cell either has a central state given initially, or it is initialized as $q_0$ when the cell is activated. Note that *grey* cells gather informations about their $t$-neighbors like *black* cells do but don't send their central state because some of their $t$-neighbors aren't marked yet and the signals might be lost.

It is important to see that the simulation of $\mathcal{A}_3$ by $\mathcal{A}_2$ is therefore asynchronous because the colored cells apply the transition function of $\mathcal{A}_3$ at different times. This means that the number of $\mathcal{A}_3$ generations computed by the colored cells isn't the same for all. However, there is a dependency between a cell and its $t$-neighbors that ensures that, at a given time $\tau$, each colored cell $t(c)$ has computed at most one generation more than any of its $t$-neighbors. This gives a sort of *partial-synchronicity* and proves that a *black* or *grey* cell can never receive more than two consecutive signals from any of its $t$-neighbors before it can process them, and so each cell can have only a finite memory.

This automaton simulates $\mathcal{A}_3$ in the sense that for a given configuration $\mathcal{C}_3$, for any cell $c \in \mathbb{Z}^3$, the sequence of states of $c$ in the evolution of $\mathcal{A}_3$ starting on $\mathcal{C}_3$ is exactly the sequence of central states of $t(c)$ restricted to the times when $t(c)$ is *red*.

The time needed for a given cell $t(c)$ to compute $k$ generations of its corresponding cell $c$ is at most the time needed to activate all the image by $t$ of the sphere of radius $k$ centered on $c$, plus the radius of the image of this sphere (it's the time needed for the furthest information to reach $t(c)$). We have seen that these spheres grow polynomially, and that they are constructed in polynomial time so for every cell $c \in \mathbb{Z}^3$ there exists a polynomial $P_c \in \mathbb{Z}[X]$ of degree at most 4 such that $t(c)$ has computed $k$ generations of $c$ in time at most $P_c(k)$. Note that the polynomial depends on $c$ but considering the relative sizes of the

spheres of radius $k$ in $\mathbb{Z}^3$ and $\mathbb{Z}^2$ it is easy to prove that we cannot simulate $k$ generations of $\mathcal{A}_3$ on $\mathbb{Z}^2$ in a time bounded by a global polynomial (it is in fact impossible to bound it by any function).

## 5 Conclusion

What we have done here can easily be done to simulate any $n$-dimensional CA on a 2DCA. However doing such a construction to simulate a 2DCA with a 1DCA doesn't work (mainly because we cannot embed as we did the 2D neighborhood in a 1D neighborhood and use a similar construction without important conflicts).

Many things can now be studied about this simulation. For example, the $f$, $g$ and $h$ functions used here are bounded by polynomials of degree 3. Considering the size of the spheres, we see that it is not possible to have polynomials of degree lower than $3/2$ but where exactly is the real lower bound of the simulation? Also the irregularity of the functions that we use here make them hard to compute with a 2DCA, but having a better understanding of their growth (a good lower bound) or finding simpler functions (still bounded polynomially) could highly improve the construction in terms of time and number of necessary states.

The automaton constructs a representation of $\mathbb{Z}^3$ in $\mathbb{Z}^2$. It would be interesting to study how usual 3-dimensional objects are represented, and if natural 3-dimensional cellular automata configurations can be identified easily after projection.

We have seen here how to encode finite configurations of $\mathcal{A}_3$ as finite configurations of $\mathcal{A}_2$. Periodic $\mathcal{A}_3$ configurations can also be simulated by a 2DCA using a method similar to the one described if we encode them as finite configurations of $\mathcal{A}_2$ using specific state markers for the borderline and sending the signals from one border to the other (as on a torus).

## References


1. Frisch, U., d'Humières, D., Hasslacher, B., Lallemand, P., Pomeau, Y., Rivet, J.P.: Lattice gas hydrodynamics in two and three dimension. Complex Systems **1** (1987) 649–707 Reprinted in *Lattice Gas Methods for Partial Differential Equations*, ed. G. Doolen, p.77, Addison-Wesley, 1990.
2. Róka, Z.: Simulations between cellular automata on Cayley graphs. Theoretical Computer Science **225** (1999) 81–111
3. Martin, B.: A geometrical hierarchy on graphs via cellular automata. Fundamenta Informaticae **52** (2002) 157–181
4. Cole, S.N.: Real-time computation by $n$-dimensional iterative arrays of finite-state machines. IEEE Transactions on Computers **C-18** (1969) 349–365