Two-dimensional cellular automata and deterministic on-line tessellation automata

Véronique Terrier
GREYC, Campus 2, Bd Maréchal Juin, 14032 Caen Cedex, France

Received 03 October 2001; received in revised form 03 July 2002; accepted 04 July 2002
Communicated by B. Durand

Abstract

In this work we consider the relationships between the classes of two-dimensional languages defined by deterministic on-line tessellation automata and by real time two-dimensional cellular automata with Moore and Von Neumann neighborhood. We generalize the result known for one-dimensional cellular automata to two-dimensional cellular automata with Von Neumann neighborhood: the class of real time cellular automata is closed under rotation of 180° if and only if real time cellular automata is equivalent to linear time cellular automata.

© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Cellular automata; Deterministic on-line tessellation automata; Real time; Linear time; Closure property

1. Introduction

Cellular automata appear to be a relevant model of massively parallel computation. In this paper we investigate properties of two-dimensional CA as language recognizer with parallel input mode and bounded computation. The study of the power of CA naturally leads to the comparison with other devices. Links between languages of pictures (but as linear description of pictures) and CA have been studied [10]. Here, we are interested in two-dimensional languages and so far with devices operating on two-dimensional languages. Different approaches generalize the notion of rational languages to two-dimensional pictures [3]: tiling systems, four-way automata, on-line tessellation
automata, etc. All these models restricted to one row operate as finite automata, finite automata which are less powerful than real time one-dimensional CA. Thus languages, as for instance \( \{0^n1^n : n \geq 0\} \), which separate real time CA and FA in one-dimensional case separate also real time two-dimensional CA and these two-dimensional devices. On the other hand, the inclusions properties do not hold automatically; for these devices, unlike one-dimensional FA, the non-deterministic versions are more powerful than the deterministic ones. Kosaraju [7] has set that any language recognized by four-way automata with one pebble can be recognized in linear time by two-dimensional CA. For tiling systems where non-determinism is involved, their relationship with two-dimensional CA seems not to be obvious. So we restrict our investigation to deterministic on-line tessellation automata (DOTA) introduced by Inoue and Nakamura [5] as a special type of two-dimensional CA. Ito and Inoue [6] have shown that DOTA are equivalent to two-way two-dimensional alternating finite automata through 180° rotation.

We will see that two-dimensional CA with Von Neumann neighborhood as well with Moore neighborhood can “simulate” in real time the computation of any DOTA. Two-dimensional CA with Von Neumann neighborhood can also recognize in real time DOTA languages through a rotation of 180°. On the other hand, there exists a DOTA language whose rotation of 180° is not real time recognizable on a two-dimensional CA with Moore neighborhood. For non-deterministic on-line tessellation automata (OTA), note that it is incomparable with two-dimensional CA with Moore neighborhood; but we do not know whether it is less powerful than two-dimensional CA with Von Neumann neighborhood.

So for Moore neighborhood the way of supplying the input influences the recognition time. If we consider Von Neumann neighborhood we do not know if it is closed under rotation of 180°. But as Ibarra and Jiang [4] have shown in one-dimensional case, the answer is yes if and only if real time cellular automata is equivalent to linear time cellular automata.

The paper is organized as follows. In Section 2, we recall some basic definitions. Then we compare on-line tessellation automata to two-dimensional CA with Moore neighborhood in Section 3 and with Von Neumann neighborhood in Section 4. Section 5 establishes, for Von Neumann neighborhood, the equivalence between the question of equality of linear time and real time and the question of closure under rotation of 180°.

### 2. Definitions

A picture \( p \) of size \((m,n)\) over an alphabet \( \Sigma \) is an \( m \times n \) array of elements of \( \Sigma \) with \( p(i,j) \) (where \( 1 \leq i \leq m \) and \( 1 \leq j \leq n \)) being the symbol on the \( i \)th row and the \( j \)th column of \( p \):

\[
p = \begin{bmatrix}
p(1,1) & \ldots & p(1,n) \\
\vdots & \ddots & \vdots \\
p(m,1) & \ldots & p(m,n) 
\end{bmatrix}
\]
The rotation of $180^\circ$ of the picture $p$ is

$$p^{180} = \begin{vmatrix} p(m,n) & \cdots & p(m,1) \\ \vdots & \ddots & \vdots \\ p(1,n) & \cdots & p(1,1) \end{vmatrix}$$

$L^{180}$ denotes the rotation of a language $L$.

The $t$th diagonal $D_t$ of an $m \times n$ array of cells is \{$(i,j): i+j=t+1$, $1 \leq i \leq m$, $1 \leq j \leq n$\}.

A DOTA is specified by the 5-tuple $(\Sigma, S, S_{\text{accept}}, \delta, \#)$ where $\Sigma$ is the input alphabet, $S$ the set of states, $S_{\text{accept}} \subseteq S$ the set of accepting states, $\#$ the boundary symbol and $\delta$ the transition function from $S^2 \times \Sigma$ into $S$ (see Fig. 1). A DOTA is composed of a two-dimensional rectangular array of cells indexed by $N^2$. Initially, the symbol $p(i,j)$ of the input picture $p$ is stored on the cell $c(i,j)$. A run on the picture $p$ consists in computing on each cell $c(i,j)$ a state denoted $q(i,j)$ according to the transition function $\delta$ and the symbol $p(i,j)$, the states $q(i,j-1)$ and $q(i-1,j)$ of its west and north neighbors: $q(i,j) = \delta(q(i,j-1), q(i-1,j), p(i,j))$. The cells with no west or north neighbors use by default the boundary state $\#$ as neighbor’s state. The computation is not totally parallel, it flows down one diagonal at a time. It starts at time 1 on the cell $c(1,1)$ which enters the state $q(1,1) = \delta(\#, \#, p(1,1))$. At time $t$ the cells of the $t$th diagonal $D_t$ evolve simultaneously: each cell $c(x,1+t-x)$ enters a state $q(x,1+t-x) = \delta(q(x,t-x), q(x-1,1+t-x), p(x,1+t-x))$. It ends at time $m+n-1$ on the cell $c(m,n)$. The DOTA accepts the input picture $p$ if the state $q(m,n)$ is an accepting state.

A two-dimensional cellular automata $A$ is a 5-tuple $(\Sigma, S, S_{\text{accept}}, N, \delta)$ where $\Sigma \subseteq S$ is the input alphabet, $S$ the set of states, $S_{\text{accept}} \subseteq S$ the set of accepting states, $N \subseteq \mathbb{Z}^2$ the neighborhood of size $k$, $\delta: S^k \rightarrow S$ the transition function. A two-dimensional cellular automata is composed of a two-dimensional array of cells which evolve...
synchronously at discrete time steps according to its neighborhood. We deal with two different neighborhoods: the Von Neumann neighborhood \( N_{VN} = \{ (x, y) : \| (x, y) \|_1 \leq 1 \} \) (where \( \| (x, y) \|_1 = |x| + |y| \)) and the Moore neighborhood \( N_{Moore} = \{ (x, y) : \| (x, y) \|_\infty \leq 1 \} \) (where \( \| (x, y) \|_\infty = \max(|x|, |y|) \)). We consider only parallel input mode: at time 0 the symbol \( p(i, j) \) of the input picture \( p \) is stored on the cell \( c(i, j) \). We assume that the computation is bounded: the other cells remain in a special state, the boundary state #, during all the computation. The state of the cell \( c(i, j) \) at time \( t \) (denoted \( \langle i, j, t \rangle \)) is computed according to the transition function \( \delta \) and the states of its neighborhood at previous time. For instance, for a CA with Von Neumann neighborhood, we have initially \( \langle i, j, 0 \rangle = p(i, j) \) and for steps \( t \geq 1 \), \( \langle i, j, t \rangle = \delta(\langle i, j, t-1 \rangle, \langle i+1, j, t-1 \rangle, \langle i, j+1, t-1 \rangle, \langle i-1, j, t-1 \rangle, \langle i, j-1, t-1 \rangle) \).

We say that a CA accepts the picture \( p \) in \( t \) steps if the upper left cell \( c(1, 1) \) enters an accepting state at time \( t \). Let \( T \) be a function from \( N^2 \) into \( N \). If for any picture \( p \) of size \( (m, n) \) the CA can determine whether \( p \in L \) within \( T(m, n) \) steps then the CA is said to recognize \( L \) in time \( T \). Real time refers to the minimal time needed by the upper left cell \( c(1, 1) \) to read any particular part of the input; it corresponds modulo a constant to the function \( T(m, n) = \max(m, n) - 1 \) for Moore neighborhood and the function \( T(m, n) = m + n - 2 \) for Von Neumann neighborhood. And linear time refers to \( T(m, n) \leq c \max(m, n) \) for Moore neighborhood (resp. \( T(m, n) \leq c(m + n) \) for Von Neumann neighborhood) where \( c > 1 \) is any constant.

The classes of languages recognized by deterministic on-line tessellation automata are denoted by \( L(DOTA) \). \( L^{180}(DOTA) \) is the class of languages \( L^{180} \) rotations of languages \( L \) of \( L(DOTA) \). The classes of languages recognized in real time (resp. linear time) by two-dimensional cellular automata with Moore and Von Neumann neighborhood are denoted by \( RPCA_{Moore} \) and \( RPCA_{VN} \) (resp. \( LPCA_{Moore} \) and \( LPCA_{VN} \)).

3. CA with Moore neighborhood and DOTA

Fact 1. For any DOTA there exists an RPCA\textsubscript{Moore} that “simulates” it.

Proof. See Fig. 2. Without loss of generality, we suppose that the inputs size \( (m, n) \) is such that \( m \leq n \), the case \( m > n \) is symmetric. Let \( M \) be the trace of a signal initiated at time 1 which runs diagonally and is reflected on the north and south borders. In defining the function \( f \) by

\[
f(j) = \begin{cases} 
  j \mod (m-1) & \text{if } j \ \text{div} (m-1) \text{ is even}, \\
  m - 1 - (j \mod (m-1)) & \text{else},
\end{cases}
\]

the part of \( M \) which is characterized at time \( t \) will correspond to \( \{ (1 + f(j), 1 + j) : j = 0, \ldots, t-1 \} \). We will see how the computation achieved on the diagonals \( D_{2j-1} \) and \( D_{2j} \) of the DOTA will be simulated on the cells \( \{ (1 + f(j), 1 + j) : \max(0, t-m) \leq j < t \} \) at time \( t \). We first describe the move of the data towards \( M \). For that purpose at time 1, each symbol \( p(i, j) \) is sent at maximal speed to the northwest. When \( i > j \) it is reflected on the west border, then it runs to the northeast. When \( i < j \) it is reflected on the north border, then it runs to the southwest, after \( m-1 \) steps it is
In particular if \( m \) reaches the cell \( c_{i,j} \) on the south border, then it runs to the northwest and so on. When \( i = j \) it reaches the cell \( c(1,1) \).

In this way, we could verify that:

- If \( i \geq t \), the symbol \( p(i,2t-i) \) which is on \( D_{2t-1} \) at initial time will reach the cell \( c(1+i-t,1+i-t) \) at time \( t-1 \) and the symbol \( p(i,2t+1-i) \) which is on \( D_{2t} \) at initial time will reach the cell \( c(2+i-t,1+i-t) \) at time \( t-1 \).
- If \( i \leq t \), \( p(i,2t-i) \) will reach the cell \( c(1+f(t-i),1+t-i) \) at time \( t-1 \); and \( p(i,2t+1-i) \) will reach the cell \( c(1+f(t-i),2+t-i) \) at time \( t-1 \).

Actually, the input symbols of the first row and the first column must move without delay to the northeast and the southwest. But as initial time no cell knows whether it is on border or not, all symbols are sent both to the northeast and the southwest. Thus, useless signals are generated but they will not interfere in the computation.

To complete the proof we now describe the computation on the diagonal \( M \). We could show by induction that the CA have all the required information to compute:

- At time \( t=1 \), the values \( q(1,1) \), \( q(1,2) \), \( q(2,1) \) on the cell \( c(1,1) \).
- At time \( t<i \),
  - \( q(i,2t-i) \) in the cells \( c(1+i-t,1+i-t) \) and \( c(i-t,i-t) \), as at previous time the cell \( c(1+i-t,1+i-t) \) contains \( p(i,2t-i) \) and \( q(i-1,2t-i) \) and the cell \( c(i-t,i-t) \) contains \( q(i,2t-i-1) \).
  - \( q(i,2t+1-i) \) in the cell \( c(1+i-t,1+i-t) \) as at previous time the cell \( c(2+i-t,1+i-t) \) contains \( p(i,2t+1-i) \) and at the same time \( t \), \( q(i,2t-i) \) and \( q(i-1,2t+1-i) \) are computed in the same cell \( c(1+i-t,1+i-t) \).
- At time \( t \geq i \),
  - \( q(i,2t-i) \) in the cells \( c(1+f(t-i),1+t-i) \) and \( c(1+f(t-1-i),t-i) \), as at previous time the cell \( c(1+f(t-i),1+t-i) \) contains \( p(i,2t-i) \) and \( q(i-1,2t-i) \) and the cell \( c(1+f(t-1-i),t-i) \) contains \( q(i,2t-i-1) \).
  - \( q(i,2t+1-i) \) in the cell \( c(1+f(t-i),1+t-i) \) as at previous time the cell \( c(1+f(t-i),2+t-i) \) contains \( p(i,2t+1-i) \) and at the same time \( t \), \( q(i,2t-i) \) and \( q(i-1,2t+1-i) \) are computed in the same cell \( c(1+f(t-i),1+t-i) \).

In particular if \( m+n \) is even, \( q(m,n) \) is known at time \( (m+n)/2 \) in \( c(1+f((n-m)/2-1),(n-m)/2) \) and additional \( \text{max}(f((n-m)/2-1),(n-m)/2-1) = (n-m)/2-1 \).
steps are required to reach the cell $c(1,1)$. And if $m+n$ is odd, $q(m,n)$ is known at time $(m+n−1)/2$ in $c(1+f((n−m−1)/2),(n−m+1)/2)$ and additional $\max(f((n−m−1)/2),(n−m+1)/2−1)=(n−m−1)/2$ steps are required to reach the cell $c(1,1)$. So in both cases the result is known on the cell $c(1,1)$ at time $n−1$.

Let $L$ be the language $\{ p \in \{0,1,\}^* : p$ is of size $(m,n)$ with $m \geq 2(\log_2 n+2) \quad n \geq 2(\log_2 m+2)$ and there exists $i,j$ such that $1 \leq i \leq m/2, \quad 1 \leq j \leq n/2, \quad p(i,j) = 1, \quad p(m,\lfloor n/2 \rfloor) = \cdots = p(m,n−\lfloor \log(m−i) \rfloor−2) = 0, \quad p(m,n−\lfloor \log(m−i) \rfloor−1) \cdots p(m,n−1)$ is the binary notation of $n−j$ and $p(m,\lfloor n/2 \rfloor) = \cdots = p(m−\lfloor \log(n−j) \rfloor−2,n) = 0, \quad p(m−\lfloor \log(n−j) \rfloor−1,n) \cdots p(m−1,n)$ is the binary notation of $n−j$}. Roughly describing, the pictures $p$ of size $(m,n)$ belonging to $L$ are such that the last line contains the binary notation of $m−i$, the last column contains the binary notation of $n−j$ and $p(i,j) = 1$ (Fig. 3).

**Fact 2.** The language $L$ belongs to $L^{180}(DOTA)\setminus RPCA_{Moore}$.

**Proof.** In [11] it has been shown that $L$ belongs to $RPCA_{VN} \setminus RPCA_{Moore}$. Note that the computation on this $RPCA_{VN}$ is a wave which runs diagonally. Hence the same process can be used on a DOTA to recognize the language $L^{180}$. Then $L$ belongs to $L^{180}(DOTA)\setminus RPCA_{Moore}$. □

Regarding a DOTA operates as a finite automaton on pictures of size $(1,n)$ or $(m,1)$ and the previous facts, the following proposition holds:

**Proposition 1.** 1. $L(DOTA) \not\subseteq RPCA_{Moore}$;
2. $L^{180}(DOTA)$ is incomparable with $RPCA_{Moore}$;
3. $RPCA_{Moore}$ is not closed under rotation of $180^\circ$.

4. **CA with Von Neumann neighborhood and DOTA**

**Fact 3.** For any $DOTA$ there exists an $RPCA_{VN}$ that “simulates” it.

**Proof.** See Fig. 4. Such an $RPCA_{VN}$ will simulate at time $t−1$ on its first row the computation achieved on the diagonal $D_t$ of the $DOTA$. 

Fig. 3. The language $L$. 
We first describe the move of the data towards the first row, more precisely how the data \( p(i,j) \) of the diagonal \( D_{i+j-1} \) reaches at time \( i+j-2 \) the cell \( c(1,j) \) when \( i+j \leq m \) and the cell \( c(1,m+1-i) \) when \( i+j > m \).

Let \( W_k = \{ c(1,k), \ldots, c(m-k,k) \} \) be the cells at distance \( k \) from the west border and at distance greater than \( k \) from the south border, and \( S_k = \{ c(m+1-k,k), \ldots, c(m+1-k,n) \} \) be the cells at distance \( k \) from the south border and at distance greater or equal to \( k \) from the west border. At time 1 from each west (resp. south) border cell a signal is sent to the east (resp. north). These signals run at maximal speed and hence are able to characterize the cells belonging to \( W_k \) (resp. \( S_k \)) at time \( k \). From this time \( k \) each cell of \( W_k \) (resp. \( S_k \)) records the content of its south (resp. east) neighbor. In this way the data \( p(i,j) \),

- if \( i+j \leq m+1 \), starts from step \( j \) to move to the north and hence reaches the cell \( c(1,j) \) at time \( i+j-2 \);
- if \( i+j > m+1 \), starts from step \( m+1-i \) to move to the west, reaches the diagonal \( D_m \) on the cell \( c(i,m+1-i) \) at time \( j-1 \), then moves to the north and hence reaches the cell \( c(1,m+1-i) \) at time \( i+j-2 \).

To complete the proof we now describe the computation on the first row. We could show by induction that the CA have all the required information to compute

- \( q(1,j) \) and \( q(2,j) \) for \( j \) such that \( 1 \leq j < m \) in the cell \( c(1,j) \) at time \( j \), as at previous time the cell \( c(1,j) \) contains \( p(1,j) \), its west neighbors \( q(1,j-1) \) and \( q(2,j-1) \), its north neighbor \( \# \) and its south neighbor \( p(2,j) \);
- \( q(i,j) \) for \( i,j \) such that \( i > 2 \) and \( i+j \leq m+1 \) in the cell \( c(1,j) \) at time \( i+j-2 \), as at previous time the cell \( c(1,j) \) contains \( q(i-1,j) \), its west neighbor \( q(i,j-1) \) and its south neighbor \( p(i,j) \).

And from time \( m \) (notified on the first row by the north signals initiated on the south border cells at time 1)
• $q(1,m)$ and $q(1,m+1)$ in the cell $c(1,m)$ at time $m$, as at previous time the cell $c(1,m)$ contains $p(1,m)$, its west neighbor $q(1,m-1)$, its north neighbor # and its east neighbor $p(1,m+1)$;
• $q(i,j)$ in the cell $c(i,m+1-j)$ at time $i+j-2$ for $i,j$ such that $i+j\geq m+1$ as at previous time the cell $c(i,m+1-j)$ contains $q(i,j-1)$, its east neighbor $q(i-1,j)$ and its south neighbor $p(i,j)$ (except for $i = 1$ its east one).

In this way the state $q(m,n)$ is computed on the cell $c(1,1)$ at time $m+n-2$. □

The following fact is straightforward.

**Fact 4.** $L_{180}(DOTA)$ is included in $RPCA_{VN}$.

**Proof.** Rotate the computation of the $DOTA$ by $180^\circ$ to get this one of the $RPCA_{VN}$. □

The following proposition sums up the consequences of the previous facts.

**Proposition 2.** 1. $L(DOTA) \subseteq RPCA_{VN}$;
2. $L_{180}(DOTA) \not\subseteq RPCA_{VN}$.

5. Closure under rotation of $180^\circ$ for CA with Von Neumann neighborhood

In Section 3 we have seen that $RPCA_{Moore}$ is not closed under rotation of $180^\circ$. What about $RPCA_{VN}$? Observe that the rotation of $180^\circ$ for pictures restricted to one row coincides to the reversal operation. For one-dimensional case Ibarra and Jiang [4] have established that the class of real time CA is closed under reversal if and only if real time CA is equivalent to linear time CA. Actually, this result can be extended to the two-dimensional case: for Von Neumann neighborhood the question of closure under rotation corresponds to the question of equivalence between real time and linear time.

**Proposition 3.** $RPCA_{VN} = LPCA_{VN}$ if and only if $RPCA_{VN}$ is closed under rotation of $180^\circ$.

**Proof.** The proof method is as in [2,4]. The “only if” part is straightforward as $LPCA_{VN}$ is closed under rotation of $180^\circ$. Conversely, the scheme of the proof is the following. Let $L_1$ be a language belonging to $LPCA_{VN}$. We suppose without loss of generality that the pictures of $L_1$ are of size $(m,n)$ with $m \leq n$. Indeed we could always deal in one hand with $L_1 \cap \{ p \in \{ \Sigma \}^{**} : p \text{ of size } (m,n) \text{ and } m \leq n \}$ and in another hand with $L_1 \cap \{ p \in \{ \Sigma \}^{**} : p \text{ of size } (m,n) \text{ and } m > n \}$ in a symmetric manner. From $L_1$ we define a new language $L_2 = \{ p \oplus r : p \in L_1, r \in \{ \$ \}^{**}, p \text{ of size } (m,n), r \text{ of size } (m,n') \text{ with } n' = 2^{[\log((m+n)/2)]} \}$ (where $\oplus$ stands for the column concatenation). We will show in Fact 5 that $L_2 \in RPCA_{VN}$. Since by hypothesis $RPCA_{VN}$ is closed under
rotation of 180°, we get \( L_2^{180} \in RPCAV_N \). Then it follows, as it will be shown in Fact 6, that \( L_1^{180} \in RPCAV_N \). Finally with the use of hypothesis, we get \( L_1 \in RPCAV_N \).

It remains to establish the two used Facts 5 and 6.

**Fact 5.** If \( L_1 \in LPCAV_N \) then \( L_2 = \{ p \oplus r : p \in L_1, r \in \{ $ \}^* \} \) of size \((m,n), r \) of size \((m,n')\) with \( n' = 2^{\lfloor \log((m+n)/2) \rfloor} \) \( \in RPCAV_N \).

**Proof.** \( L_1 \in LPCAV_N \), so it is recognized in time \( m + n + u(m+n) \) for some constant \( u > 0 \). Then by linear acceleration (presented in [11]), choosing a constant \( v > 4u \), \( L_1 \) is recognized by a CA \( A \) in time \( m + n + (u/v)(m+n) < m + n + (m+n)/4 \), in other words before the time \( m + n + 2^{\lfloor \log((m+n)/2) \rfloor} - 2 \). So on an input picture a CA can

- check if the picture is of shape \( \Sigma^* \cup \{ \$ \}^* \) in real time;
- check on the way constituted by the first column and the last row if it is of shape \( a_1 \ldots a_m \ldots a_n \$'s \) with \( n' = 2^{\lfloor \log((m+n)/2) \rfloor} \) and get the result on cell \( c(1,1) \) at time \( m + n + n' - 2 \);
- simulate the CA \( A \) on the input \( p \) of size \((m,n)\) which is on the left part surrounding in the north, west and south by \# and in the east by \$ and so it enters an accepting or rejecting state on cell \( c(1,1) \) before the time \( m + n + n' - 2 \) depending on whether \( p \) belongs or not to \( L_1 \);
- send an end signal initiated on cell \( c(m-1,n+n'-1) \) distinguishable at time 2 towards the cell \( c(1,1) \) to mark it at time \( m + n + n' - 2 \).

Hence the CA is able to collect all the results at time \( m + n + n' - 2 \).

**Fact 6.** If \( L_2^{180} \in RPCAV_N \) then \( L_1^{180} \in RPCAV_N \).

**Proof.** The outline of the proof is the same than in dimension 1 [2]. Let \( \mathcal{A}_2 \) be a CA which recognizes in real time \( L_2^{180} \). To test if a picture \( p \) of size \((m,n)\) belongs to \( L_1^{180} \), in compressing the input, we will add in the west a pack \( r \) of symbols \$ of size \((m,n')\) with \( n' = 2^{\lfloor \log((m+n)/2) \rfloor} \). Then it will be sufficient to simulate the automaton \( A_2 \) on this input \( r \oplus p \). However, \( n' \) the width of the \$’s pack \( r \) depends on the input size \((m,n)\). But the cells of the array have the knowledge of this size when they get signals from the different borders and it is too late. So by default we will add \$’s packs \( r_k \) of width \( n' = 2^k \) to the picture \( p \) for \( k = 1, 2, \ldots, \log[(m+n)/2] \) and of height lower than \( 2^{k+1} \) as we deal with pictures of size \((m,n)\) where \( m \leq n \). In other words, we will simulate the CA \( \mathcal{A}_2 \) on the inputs \( r_k \oplus p|_{Z_k} \) for \( k = 1, \ldots, \log[(m+n)/2] \) with \( r_k \) of size \((\min(m,2^{k+1}-1),2^k)\) and \( r(i,j) = \$ \) and with \( p|_{Z_k} \) the part of the picture \( p \) delimited by \( Z_k = \{(x,y) : 1 \leq x < \min(m,2^{k+1}) - 1, y \geq 1 \} \) and \( x + y < 2^{k+2} \). Practically, we have to describe where and how the different inputs \( r_k \oplus p|_{Z_k} \) are set up. It is useful to recall how it is performed in dimension 1. See Fig. 5 and [2]. First the symbols of the input word \( p = a_1 \ldots a_n \) are grouped two by two on the diagonal \( t = c + 1 \) (the cell \( c \) at time \( c + 1 \) contains \( a_{2c-1} \) and \( a_{2c} \)). Second by means of signals the curves \( S_k = A_k \cup C_k \cup E_k \), where \( A_k = \{(c,c+2^{k-1}) : 1 \leq c < 2^{k-2} \}, C_k = \{(c,c+2^{k-1}) : 1 + 2^{k-2} \leq c < 2^{k-2} + 2^{k-1} \} \) and \( E_k = \{(c,3c - 2^k - 2) : 1 + 2^{k-2} + 2^{k-1} \leq c < 2^{k-2} + 2^{k} \} \), are characterized; and in order to set up \$’s \( a_1 \ldots a_{\min(n,2^{k+2})} \) on \( S_k \), \$’s are introduced and the input symbols are
moved and grouped again two by two. So 4$ by cell are set up on the \( 2^{k-2} \) cells of \( A_k \), the symbols \( a_1, \ldots, a_{2^{k+1}} \) are grouped by 4 on the \( 2^{k-1} \) cells of \( C_k \) and the symbols \( a_{1+2^{k+1}}, \ldots, a_{\min(n,2^{k+2})} \) are grouped by 4 on the \( 2^{k-1} \) cells of \( E_k \). Afterwards the simulation of \( \mathcal{A}_2 \) on \( S_k \), \( a_1 \ldots a_{\min(n,2^{k+2})} \) takes place on the area delimited below by \( S_k \) and up by the line \( c+t = \min(n,2^{k+2}) \). Let us come back to dimension 2. Relating the grouping, with Von Neumann neighborhood it is done according to the diagonals, see Fig. 6. As in dimension 1 a first grouping process is performed now in grouping the first diagonals and the second diagonals two by two; in addition, for parity reason, we group also by two according to the row and then only work on the cells \( c(i,j) \) with \( i+j \) of even parity. So we will obtain groups of height input symbols on the plan \{ \( (a,b,a+b-1) : a+b \text{ even} \} \). The second grouping process groups again two by two according to the first and second diagonals. So the symbols of input \( p \) will be grouped by 32, moreover the grouped cells’ neighborhood of radius 2 contains the elementary neighborhood of radius 8 which further will allow simulating four steps of \( \mathcal{A}_2 \) in one step.

The first grouping process is described in point 1. In point 2, we show how to characterize some surfaces \( S_k \) where \( r_k \oplus p|_{Z_k} \) will be set up and in points 3, 4, 5, how the input symbols \( p(x,y) \) are compressed and moved towards the surface \( S_k \). Then in point 6, we will show how to simulate the CA \( \mathcal{A}_2 \) with input \( r_k \oplus p|_{Z_k} \) supplied on the surface \( S_k \). Finally in point 7, we will insure that the different simulations do not overlap more than three times.

1. Grouping process towards the plane \{ \( (a,b,a+b-1) : a+b \text{ even} \} \), see Fig. 7.

On the cell \( c(a,b) \) at time \( a+b-1 \) we will group the eight input symbols \( p(x,y) \) such that \( (a+b)/2 = [(x+y)/4] + 1 \) and \( (a-b)/2 = [(x-y+1)/4] \). We describe the move of the symbols \( p(x,y) \) for \( y \geq x \). The move of the symbols \( p(x,y) \) for \( y < x \) is symmetric. We will use the two following filters which can be easily set up:

- the family of rows \( R_k = \{ (x,k,2k) : x = 1, \ldots, n \} \);
- the family of diagonals \( Q_k = \{ (x,2k-x,2k-1) : x = 1, \ldots, 2k-1 \} \).

First each symbol \( p(x,y) \) is copied at time 4 on the cells

\[
\begin{align*}
  c(x-2, y-2) & \quad c(x-2, y-1) \\
  c(x-1, y-3) & \quad c(x-1, y-2) \\
  c(x-1, y-2) & \quad c(x-1, y-1) \\
  c(x, y-2) & \quad c(x, y-1)
\end{align*}
\]

From this time 4, these eight copies of \( p(x,y) \) are sent at maximal speed to the northwest. If \( x \) is odd, four of them meet the distinguished row \( x+1/2 \) at time \( x+1 \) on the columns \( y-(x+3)/2+c \) with \( c = 0,1,2,3 \); the other copies do not meet the row filter and will disappear by instance in reaching the first row. If \( x \) is even, two of them meet the distinguished row \( x/2 \) at time \( x \) on the columns \( y-x/2+c \) with \( c = 0,1 \) and two of them meet the distinguished row \( x/2+1 \) at time \( x+2 \) on the columns \( y-x/2+c \) with \( c = -1,0 \); the other ones disappear. Then from these sites they are sent at maximal speed to the west. If \( x \) is odd, one of them meet the distinguished diagonal \( Q_{\{i\oplus(x+y)/4\}+1} \) on the site \( \{ (x+1)/2,2\}[(x+y)/4] +2-(x+1)/2, 2[(x+y)/4]+1 \); the other three copies do not meet the diagonals filter. If \( x \) is even, one of them meet the distinguished
Fig. 5. In dimension 1.
Fig. 6. Grouping 8 by 8.

Fig. 7. Grouping process towards \{(a, b, a + b - 1) : a + b \text{ even}\}.
diagonal $Q_{(x+y)/4}+1$ on the site $\langle x/2, 2[(x+y)/4] + 2 - x/2, 2[(x+y)/4] + 1 \rangle$ if $x+y \equiv 2, 3 \mod(4)$ and on the site $\langle x/2 + 1, 2[(x+y)/4] + 1 - x/2, 2[(x+y)/4] + 1 \rangle$ if $x+y \equiv 0, 1 \mod(4)$. We could check that these sites $\langle a, b, a+b-1 \rangle$ verify $(a+b)/2 = [(x+y)/4] + 1$ and $(a-b)/2 = [(x-y+1)/4]$.

2. Characterization of the surfaces $S_k$ where the inputs $r_k \odot p|_{Z_k}$ will be supplied, see Fig. 8.

The cells $c(i,j)$ with $i+j$ even and $j \leq 2^{k-2}$ will contain $S$’s. The cells $c(i,j+2^{k-2})$ with $i+j$ even will group the contents of the sites $(a,b,a+b-1)$ such that $(a,b) = (2i-2, 2j-2), (2i-1, 2j-3), (2i-1, 2j-1), (2i, 2j-2)$. In other words the cells $c(i,j+2^{k-2})$ with $i+j$ even will contain the 32 input symbols $p(x,y)$ such that $(i+j)/2 = [(x+y+4)/8] + 1$ and $(i-j)/2 = [(x-y+1)/8]$. Moreover, for the pictures $p$ of size $(m,n) \in (Z_k \backslash Z_{k-1}) \cap \{(x,y): x \leq y\} = \{(x,y): 1 \leq x \leq y$ and $2^{k+1} \leq x+y < 2^{k+2}\}$, we have to realize a simulation of $\mathcal{S}_2$ on the input $r_k \odot p|_{Z_k}$ in real time. In particular, we cannot lose time during the $S$’s packing and grouping process with the input $p(m,n)$ which must be known as soon as possible on the output cell $c(1,1)$. This process sends $p(m,n)$ from the site $\langle m,n,0 \rangle$ to the cell $c(i,j+2^{k-2}) = c([(m+n+1)/4] + 1 + (m-n+1)/8, [(m+n+4)/8] + 1 - (m-n+1)/8 + 2^{k-2}) \sim c(m/4, n/4 + 2^{k-2})$; as $n \geq (m+n)/2 < 2^{k+1}$, it requires $|m-n/4 + n-n/4 - 2^{k-2}| = |3(m+n)/4 - 2^{k-2}|$ steps, i.e. we will not waste time if it arrives on the cell $c(i,j+2^{k-2})$ at time $3i+3(j+2^{k-2})-2^k$.

In the other part to avoid the overlapping of the different simulations, the simulation of $\mathcal{S}_2$ on the input $r_k \odot p|_{Z_k}$ will start after the time $2^{k-1}$. For these reasons among others, we define $S_k$ as a curved surface, union of these five planes:

$$A_k = \{(i,j,2^{k-1} + i+j-1): 1 \leq i \leq 2^{k-2}, 1 \leq j \leq 2^{k-2} \text{ and } i+j \text{ even}\}$$
$$B_k = \{(i,j,3i+j-1): 2^{k-2} + 1 \leq i \leq 1 + 2^{k-1}, 1 \leq j \leq 2^{k-2} \text{ and } i+j \text{ even}\}$$
$$C_k = \{(i,j,2^{k-1} + i+j-1): 1 \leq i \leq 2^{k-2}, 2^{k-2} + 1 \leq j, i+j \leq 2^{k-1} + 2^{k-2} \text{ and } i+j \text{ even}\},$$

Fig. 8. The surfaces $S_k$. 

\[\text{Image of the surface diagram}\]
\[ D_k = \{(i, j, 3i + j - 1) : 2^{k-2} + 1 \leq i \leq 1 + 2^{k-1}, 2^{k-2} + 1 \leq j \leq 2^{k-1} + 1 \text{ and } i + j \text{ even}\}, \]

\[ E_k = \{(i, j, 3i + 3j - 2^k - 5) : 1 \leq i \leq 1 + 2^{k-1}, 2^{k-1} + 1 \leq j, \]

\[ 2^{k-1} + 2^{k-2} + 2 \leq i + j \leq 2^k + 2^{k-2} + 2 \text{ and } i + j \text{ even}\}. \]

The different parts of \( S_k \) will be defined from the sites

\[ M_k = (1, 1, 2^{k-1} + 1), \]
\[ N_k = (2^{k-2} - 1, 2^{k-1} + 2^{k-2}), \]
\[ O_k = (1 + 2^{k-1}, 1, 3 + 2^k + 2^{k-1}), \]
\[ P_k = (1, 1 + 2^{k-2}, 1 + 2^{k-2} + 2^{k-1}) \]

and the two segments

\[ U_k = \{(i, j, 2^k + 2^{k-2} - 1) : i + j = 2^k - 1 + 2^{k-2}, 1 \leq i \leq 2^{k-2}\}, \]
\[ V_k = \{(i, j, 2^{k+1} + 2^{k-1} + 2^{k-2} + 1) : i + j = 2^k + 2^{k-2} + 2, 1 \leq i \leq 2^{k-1} + 1\}. \]

Concerning the segments, recall that in dimension 1 [8], there exists a CA which synchronizes a broken line of any length \( n \) in \( n - 1 \) steps, provided the two extremities be set initially in special states \( G_1 \) and \( G_2 \) (called the generals). So for any \( k \) the segment \( U_k \) will be characterized by a synchronization of the broken line \( \{(1, 2^k - 1, 2^{k-2} - 1), (2, 2^k - 2, 2^{k-2} - 1), \ldots, (c, 2^k - 1, 2^{k-2} - c), (c + 1, 2^k - 1, 2^{k-2} - c), \ldots, (2^{k-1}, 2^{k-2})\} \) of length \( 2^{k-1} - 1 \) initiated with two generals on the both extremities, on the sites \( U_{G_1,k} = (1, 2^k - 1, 2^{k-2} - 1, 2^{k-1} + 2^{k-2} + 1) \) and \( U_{G_2,k} = (2^{k-2}, 2^k - 1, 2^{k-1} + 2^{k-2} + 1) \).

In the same way \( V_k \) will be characterized by a synchronization initiated with two generals on the sites \( V_{G_1,k} = (1, 2^k + 2^{k-2} + 1, 2^k + 2^{k-2} + 2, 2^k + 2^{k-2} + 1 + 2^{k-1}) \) and \( V_{G_2,k} = (1 + 2^{k-1}, 1 + 2^{k-2}, 2^k + 2^{k-2} + 2, 2^k + 2^{k-2} + 1) \). Then techniques developed in dimension 1 [19], to mark such similar set of sites can be adjusted to characterize the sets \( \{M_k : k \geq 0\}, \{N_k : k \geq 0\}, \{O_k : k \geq 0\}, \{P_k : k \geq 0\}, \{U_{G_1,k} : k \geq 0\}, \{U_{G_2,k} : k \geq 0\}, \{V_{G_1,k} : k \geq 0\}, \{V_{G_2,k} : k \geq 0\} \) and further \( \{U_k : k \geq 0\}, \{V_k : k \geq 0\} \).

On each cell of \( A_k \) and \( B_k \) we set up a group of 32 $ symbols. And in the three following points we describe the move of the input symbols \( p(x, y) \) from the surface \( \{(a, b, a + b - 1) : a, b \text{ even}\} \) towards the other parts \( D_k, E_k, B_k \) of the surface \( S_k \).

3. Move of the data towards \( D_k = \{(i, j, 3i + j - 1) : 2^{k-2} + 1 \leq i \leq 2^{k-1} + 1, 2^{k-2} + 1 \leq j \leq 2^{k-1} + 1\} \), see Fig. 9.

We set up a filter \( H_k = \{(i, j, 3i + 3j - 2^k) : i, j \text{ odd}, 1 \leq j \leq 2^{k-1} + 1 \text{ and } 3 + 2^k \leq 2i + j \leq 3 + 2^{k-1} + 2^k\} \). It is done from the site \( (1 + 2^{k-1}, 1, 3 + 2^{k-1}) \) by the way of moves of two cells to the north and four cells to the east at maximal speed, moves of two cells to the east at speed \( \frac{1}{3} \) and moves of two cells to the south at speed \( \frac{1}{3} \).
On the site \((a - 1, b - 1, a + b - 1)\) with \(a, b\) even and \(a + b \equiv 0 \mod(4)\), the contents of the sites \((a - 2, b - 2, a + b - 5)\), \((a - 1, b - 3, a + b - 5)\), \((a - 1, b - 1, a + b - 3)\), \((a, b - 2, a + b - 3)\) are grouped. Then the content of \((a - 1, b - 1, a + b - 1)\) is sent to the north at maximal speed. It meets the filter \(H_k\) on the site \((a - 1, b - 1, a + b - 1) + (2 + 2^{k-2} - (a + b)/2, 0, (a + b)/2 - 2 - 2^{k-2}) = (1 + 2^{k-2} + (a - b)/2, b - 1, 3(1 + 2^{k-2} + (a - b)/2) + 3(b - 1) - 3 - 2^k)\). From this site it runs at maximal speed to the northeast and reaches \(D_k\) on the site \((1 + 2^{k-2} + (a - b)/2, b - 1, 3(a + b)/2 - 2^{k-2} - 3) + (b/2 - 1 - 2^{k-2}, 1 + 2^{k-2} - b/2, 2 + 2^{k-1} - b) = (a/2, b/2 + 2^{k-2}, 3a/2 + (b/2 + 2^{k-2}) - 1)\).
4. Move of the data towards $E_k \cap \{(i,j): j \leq 2^{k-1} + 2\}$.

The data will reach $E_k$ performing NW moves with supplementary W moves for ones up the diagonal $j = i + 2^{k-1}$ and supplementary N moves for ones below this diagonal.

- Move towards $E_k \cap \{(i,j): j - i \geq 2^{k-1} + 2\}$, see Fig. 10.

    We omit the description of setting up the IFF filter $I_k = \{(i,j,5i+j-1): j \equiv \pm 2 \mod(4), i \geq 1, j - i \geq 2^{k-1} + 2 \}$ on the site $(a-1, b-3, a+b-1)$ with $a, b$ even and $a+b \equiv 0 \mod(4)$ the contents of the sites $(a-2, b-2, a+b-5), (a-1, b-3, a+b-5), (a-1, b-1, a+b-3), (a-2, b-2, a+b-3)$ are grouped. It is sent at maximal speed to the northwest until the IFF filter $I_k$ on the site $(a-1, b-3, a+b-1) + (1 - a/2, 1 - a, 2 - a) = (a/2, b - a/2 - 2, 5a/2 + (b-a/2 - 2) - 1)$. Then it runs at maximal speed to the north and reaches $E_k$ on the site $(a/2, b - a/2 - 2, 2a + b - 3) + (0, 2^{k-2} + 2 - (b - a)/2, (b - a)/2 - 2^{k-2} - 2) = (a/2, b/2 + 2^{k-2}, 3a/2 + 3(b/2 + 2^{k-2}) - 2^{k-5})$.

- Move towards $E_k \cap \{(i,j): j - i \leq 2^{k-1}\}$, see Fig. 11.

    We omit the description of setting up the IFF filter $J_k = \{(i,j,i+5j - 2^{k+1} - 5): j \equiv \pm 2 \mod(4), i \geq 1, j - i \leq 2^{k-1}, i + j \leq 2^{k-1} + 2^{k+2} + 4 \}$ on the site $(a-2, b-2, a+b-1)$ with $a, b$ even and $a+b \equiv 0 \mod(4)$ the contents of the sites $(a-2, b-2, a+b-5), (a-1, b-3, a+b-5), (a-1, b-1, a+b-3), (a-2, b-2, a+b-3)$ are grouped. It is sent at maximal speed to the northwest until the IFF filter $J_k$ on the site $(a-2, b-2, a+b-1) + (-b/2 + 2^{k-2} + 2, -b/2 + 2^{k-2} + 2, b - 2^{k-1} - 4) = (b/2, b/2 + 2^{k-2}, b/2 + 2^{k-2}, (a-b)/2 + 2^{k-2}) + 5(b/2 + 2^{k-2} - 2^{k+1} - 5)$. Then it runs at maximal speed to the north and reaches $E_k$ on the site $(a-b)/2 + 2^{k-2}, b/2 + 2^{k-2}, a + 2b - 2^{k-1} - 5) + ((b-a)/2 - 2^{k-2} - 5, (a-b)/2 + 2^{k-2}) = (a/2, b/2 + 2^{k-2}, 3(a/2) + 3(b/2 + 2^{k-2}) - 2^{k-5})$.
5. Move of the data towards $C_k = \{(i, j, 2^{k-1} + i + j - 1) : 1 \leq i \leq 2^{k-2}, 2^{k-2} + 1 \leq j, i + j \leq 2^{k-1} + 2^{k-2}\}$.

$C_k$ will contain the data $p|_{Z_{k-1}}$. So a first group of data will come from $C_{k-1}$, a second group from $D_{k-1}$, the last group, on account of time constraint, will not come from $E_{k-1}$ but directly from the surface $\{(a, b, a + b - 1) : a + b \text{ even}\}$. 
• Move towards $C_k \cap \{(i,j,t) : 1 \leq i \leq 2^{k-1}, 2^{k-2} + 1 \leq j, \, i + j \leq 2^{k-1}\}$.
The contents of the sites $C_{k-1} = \{(i,j,2^{k-2} + i + j - 1) : 1 \leq i \leq 2^{k-3}, 2^{k-3} + 1 \leq j, \, i + j \leq 2^{k-2} + 2^{k-3}\}$ are sent at speed $\frac{1}{3}$ to the east. They reach $C_k$ on the sites $(i,j,2^{k-2} + i + j - 1) + (0,2^{k-3},3^{k-3}) = (i,j,2^{k-3},2^{k-1} + i + (j + 2^{k-3} - 1))$ with $1 \leq i \leq 2^{k-3}, 2^{k-2} + 1 \leq (j + 2^{k-3}), i + (j + 2^{k-3}) \leq 2^{k-1}$.

• Move towards $C_k \cap \{(i,j,t) : 2^{k-3} + 1 \leq i \leq 2^{k-2}, 2^{k-2} + 1 \leq j \leq 2^{k-2} + 2^{k-3} + 1\}$.
The contents of the sites $D_{k-1} = \{(i,j,3i + j - 1) : 2^{k-3} + 1 \leq i \leq 2^{k-2} + 1, 2^{k-3} + 1 \leq j \leq 2^{k-2} + 2^{k-3} + 1\}$, the last row $i = 1 + 2^{k-2}$ excluded, remain $2(2^{k-2} - i)$ steps on the same cell then they run at maximal speed to the east. They reach $C_k$ on the sites $(i,j,3i + j - 1) + (0,2^{k-2},-i) + (2^{k-3} + 2,2^{k-2} + 1) = (i,j,2^{k-3},2^{k-1} + i + (j + 2^{k-3} - 1))$ with $2^{k-3} + 1 \leq i \leq 2^{k-2}, 2^{k-2} + 1 \leq (j + 2^{k-3}) \leq 2^{k-2} + 2^{k-3} + 1$.

• Move towards $C_k \cap \{(i,j,t) : 1 \leq i \leq 2^{k-2}, 2^{k-2} + 2^{k-3} + 2 \leq j, 2^{k-2} + 2 \leq i + j \leq 2^{k-1} + 2^{k-2}\}$.
The move of the data towards this surface could be performed by similar processes as described in point 4. We do not give the details.

6. Simulation of the CA $\mathcal{A}_2$ on the input $r_k \oplus p|Z_k$.
Let $f_k(i,j,s)$ which will represent the time of the $s$th steps of computation on $c(i,j)$, be defined recursively:

$$f_k(i,j,0) = \begin{cases} 2^{k-1} + i + j - 1 & \text{on } A_k \text{ and } C_k \\ 3i + j - 1 & \text{on } B_k \text{ and } D_k \\ 3i + 3j - 2^k - 5 & \text{on } E_k \\ 0 & \text{otherwise} \end{cases}$$

$$f_k(i,j,s + 1) = 2 + \max_{|\alpha| + |\beta| = 2} (f_k(i + \alpha, j + \beta, s)).$$

First, we verify that the state of the cell $c(i,j)$ at time $f_k(i,j,s)$ can compute the following states of $\mathcal{A}_2 : (i,j,f_k(i,j,s)) = \{(x,y,t)_{\mathcal{A}_2} : (i+j)/2 = \lfloor (x+y+4)/8 \rfloor + 1, (i-j)/2 = \lfloor (x-y+1)/8 \rfloor, s = \lfloor t/8 \rfloor\}$. The grouping process described in the previous points insures that initially $\{(i,j,f_k(i,j,0)) : (i+j)/2 = \lfloor (x+y+4)/8 \rfloor + 1, (i-j)/2 = \lfloor (x-y+1)/8 \rfloor\}$. Now suppose that immediately after its computation the state $(i,j,f_k(i,j,s))$ is sent to all its neighbors $c(i + \alpha, j + \beta)$ where $|\alpha| + |\beta| = 2$, the cell $c(i,j)$ will get at time $2 + \max_{|\alpha| + |\beta| = 2} (f_k(i + \alpha, j + \beta, s)) = f_k(i,j,s + 1)$ all the required information to compute $(x,y,t)_{\mathcal{A}_2}$ with $(i+j)/2 = \lfloor (x+y+4)/8 \rfloor + 1, (i-j)/2 = \lfloor (x-y+1)/8 \rfloor$ and $t = 8s + 1, \ldots, 8s + 8$. Note that as $f_k(i + \alpha, j + \beta, s + 2) > f_k(i,j,s + 1)$, the number of states of each neighbor recorded at any time in $c(i,j)$ is bounded by 2.

Second on $\mathcal{A}_2$ on the input $r \oplus p$ with $p$ of size $(m,n)$, $r$ of size $(m,n')$ with $n' = 2^{\lceil \log((m+n)/2) \rceil}$, the result of computation is got on the site $(1,1,m+n+n'-2)_{\mathcal{A}_2}$.

If $(m,n) \in Z_k$, i.e. provided $k \geq \log((m+n)/2)$, these states are simulated on the site $(1,1,f_k(1,1,\lfloor (m+n+n'-2)/8 \rfloor))$. In the other part, observe that actually

$$f_k(i,j,s + 1) = 2 + \max_{\alpha + \beta = 2s, \alpha, \beta > 0} (f_k(i + \alpha, j + \beta, s))$$

$$= 2(s + 1) + \max_{\alpha + \beta = 2(s+1), \alpha, \beta > 0} (f_k(i + \alpha, j + \beta, 0)).$$
Thus

\[ f_k(1, 1, \lceil (m + n + n' - 2)/8 \rceil) = 2 \lceil (m + n + n' - 2)/8 \rceil + \max_{x + \beta = 2 \lceil (m + n + n' - 2)/8 \rceil, x, \beta > 0} (f_k(1 + x, 1 + \beta, 0)). \]

So when \( k = \lfloor \log((m + n)/2) \rfloor \) and \( x + \beta = 2 \lceil (m + n + n' - 2)/8 \rceil \), \( (1 + x, 1 + \beta) \) is on \( E_k \) and \( f_k(1 + x, 1 + \beta, 0) = 3(2 + 2 \lceil (m + n + n' - 2)/8 \rceil) - 2^k - 5 \), in addition \( n' = 2 \lceil \log((m+n)/2) \rceil = 2^k \). In this way \( f_k(1, 1, \lceil (m + n + n' - 2)/8 \rceil) = 8[(m + n + n' - 2)/8] - 2^k + 1 \); in other words, \( m + n + c \) with \( c \) being some constant.

7. Overlapping. The simulation area of \( A_2 \) on input \( r_k \circ p|_{Z_k} \) is bounded below by \( S_k \) and above by the wave which runs diagonally at maximal speed from \( V_k = \{(i, j, 2^{k+1} + 2^{k-1} + 2^{k-2} + 1) : i + j = 2^k + 2^{k-2} + 2, 1 \leq i \leq 2^{k-1} + 1 \} \) extremity of \( S_k \) towards the cell \( c(1, 1) \). Therefore, the simulation of \( A_2 \) on input \( r_k \circ p|_{Z_k} \) begins at time \( 1 + 2^k - 1 \) and ends before the time \( 1 + 2^k + 2 \). So the number of simulation areas which overlap is bounded by three.

6. Conclusion

So the link between a closure property of real time CA and the equivalence of real time and linear time CA extends over two-dimensional CA with Von Neumann neighborhood. And also the simple question whether unrestricted time CA bounded in space is more powerful than real time CA with Von Neumann neighborhood is unanswered.

References