Two-dimensional cellular automata recognizer equipped with a path

Véronique Terrier

Journées Automates Cellulaires 2008 (Uzès), pp. 174-181

TWO-DIMENSIONAL CELLULAR AUTOMATA RECOGNIZER EQUIPPED WITH A PATH

VÉRONIQUE TERRIER

GREYC, Campus II, Université de Caen, F-14032 Caen Cedex, France

Abstract. In this paper, two-dimensional cellular automata as one-dimensional language recognizers are considered. Following the approach of M. Delorme and J. Mazoyer to embed one-dimensional words into two-dimensional array, we deal with two-dimensional cellular automata equipped with a path. In this context, we investigate regular language recognition.

1. Introduction

Cellular automata (CA) is a major model of massively parallel computation. Famous examples [1, 3] illustrate the ability of CA to distribute and synchronize the information in a very efficient way. However, to determine to which extent CA can fasten sequential computation is not simple. In this context, a lot of interest has been devoted to evaluate the computation ability of one-dimensional CA as language recognizer. A question is whether increasing the dimension allows to increase the computation ability. Actually, the combinatorial capabilities of CA become more complex and new problems arise. The first ambiguity is in the way the one-dimensional input words are fed into higher dimensional arrays. For two-dimensional array, several manners have been considered. In this paper, we follow the approach introduced by M. Delorme and J. Mazoyer in [2]. They embed the input words along a path coded in an additional layer of the two-dimensional array. They prove that such CA equipped with an Archimedian path or an Hilbert path are able to recognize in real time regular languages. Here we get rid of some obstacles in considering von Neumann neighborhood instead of Moore neighborhood and especially in assuming that all cells know the position of the output. In this context, we present below such CA equipped with a path which recognize regular languages in real time whatever the path is like. The basic features of the algorithm exploit shrinking techniques.

Key words and phrases: Two-dimensional cellular automata, language recognition, regular languages.

2. Definitions

First we recall the definitions. They are mainly following the ones introduced in [2].

Definition 2.1. A deterministic finite automaton (FA) is specified by a quintuplet $(\Sigma, Q, \delta, q_{init}, Q_{\text{accept}})$ where Σ represents the input alphabet, Q the finite set of states, $q_{init} \in Q$ the initial state, $Q_{\text{accept}} \subset Q$ the set of accepting states and $\delta : Q \times \Sigma \mapsto Q$ the transition function.
Extending the transition function to strings over Σ^*, the language recognized by a finite automaton $\mathcal{F} = (\Sigma, Q, \delta, q_{init}, Q_{\text{accept}})$ is defined as $\mathcal{L}(\mathcal{F}) = \{w \in \Sigma : \delta(q_{init}, w) \in Q_{\text{accept}}\}$.

Definition 2.2. A two-dimensional cellular automaton is a two-dimensional array of identical finite automata (cells) indexed by $\mathbb{Z}^2$. In vector notation each cell is identified to its vector position $\mathbf{c} = (x_{\mathbf{c}}, y_{\mathbf{c}})$. The communication links are finite and uniform for every cell. They are specified by a finite subset of $\mathbb{Z}^2$ called the neighborhood. Each cell takes on a value from a finite set, the set of states. All cells evolve synchronously at discrete time steps according to the states of their local neighborhood. Formally the behavior of a two-dimensional cellular automaton is specified by a triplet (S, V, δ) where S represents the set of states, $V \subset \mathbb{Z}^2$ the neighborhood of size k, $\delta : S^k \mapsto S$ the transition function. Here we will restrict the neighborhood to the von Neumann one (of size 5): $V = \left\{\mathbf{v} \in \mathbb{Z}^2 : |x_{\mathbf{v}} + y_{\mathbf{v}}| \leq 1\right\}$.

Definition 2.3. The vector notations $\mathbf{n}, \mathbf{e}, \mathbf{s}, \mathbf{w}$ will denote the four directions : the north $\mathbf{n} = (-1, 0)$, the east $\mathbf{e} = (0, 1)$, the south $\mathbf{s} = (1, 0)$ and the west $\mathbf{w} = (0, -1)$. A four-connected oriented path is any sequence of positions $\mathbf{c_1}, \cdots, \mathbf{c_n}$ in $\mathbb{Z}^2$ where any two successive positions are four-adjacent: $\mathbf{c_{i+1}} - \mathbf{c_i} \in \{\mathbf{n}, \mathbf{e}, \mathbf{s}, \mathbf{w}\}$. In addition, we will restrict to simple path. That means that repeated positions are forbidden: if $i \neq j$ then $\mathbf{c_i} \neq \mathbf{c_j}$. In particular, the path is non-looping. In the sequel, we will refer to a simple and four-connected oriented path as simply a path.

Definition 2.4. A CA equipped with a path p is a CA with an additional layer which records the path p. Formally, it is specified by a quadruplet (P, S, V, δ) with $P = \{\mathbf{n}, \mathbf{s}, \mathbf{e}, \mathbf{w}, \sharp\} \times \{\mathbf{n}, \mathbf{s}, \mathbf{e}, \mathbf{w}, \sharp\} \setminus \{(\mathbf{n}, \mathbf{s}), (\mathbf{s}, \mathbf{n}), (\mathbf{e}, \mathbf{w}), (\mathbf{w}, \mathbf{e})\}$ the set of the symbols recording the path, S the set of the states, V the von Neumann neighborhood and $\delta : (S \times P)^5 \mapsto S$ the transition function. Given $p = (\mathbf{c_1}, \cdots, \mathbf{c_n})$, on the additional layer, the symbol recorded at the cell $\mathbf{c}$ will be $(\sharp, \sharp)$ if $\mathbf{c} \notin p$, $(\sharp, \mathbf{c_2} - \mathbf{c_1})$ if $\mathbf{c} = \mathbf{c_1}$, $(\mathbf{c_n} - \mathbf{c_{n-1}}, \sharp)$ if $\mathbf{c} = \mathbf{c_n}$ or $(\mathbf{c_i} - \mathbf{c_{i-1}}, \mathbf{c_{i+1}} - \mathbf{c_i})$ otherwise. It indicates how the path enters and exits the cell $\mathbf{c}$.

Definition 2.5. To specify language recognition by CA, we need to distinguish three subsets of the set of states S: Σ the input alphabet, S_{accept} the set of accepting states and S_{reject} the set of rejecting states. We also identify the cell $\mathbf{0} = (0, 0)$ as the output cell which determines the acceptance. And we have to precise the input mode. For a CA with a path, the input word is fed in parallel along this path: the i-th symbol of the input word is gotten on the i-th position of the path. Precisely, given the input word $w = w_1 \cdots w_n \in \Sigma^*$ and the path $p = (\mathbf{c_1}, \cdots, \mathbf{c_n})$, the cells are set up at initial time in these states:

$$\langle \mathbf{c}, 0 \rangle = \begin{cases} ((\sharp, \sharp), \varepsilon) & \text{if } \mathbf{c} \notin p \\ ((\sharp, \mathbf{c_2} - \mathbf{c_1}), w_1) & \text{if } \mathbf{c} = \mathbf{c_1} \\ ((\mathbf{c_n} - \mathbf{c_{n-1}}, \sharp), w_n) & \text{if } \mathbf{c} = \mathbf{c_n} \\ ((\mathbf{c_i} - \mathbf{c_{i-1}}, \mathbf{c_{i+1}} - \mathbf{c_i}), w_i) & \text{otherwise} \end{cases}$$

We say that L is recognized by a CA $\mathcal{A}$ with a path p if on input $w \in \Sigma^*$, the output cell enters an accepting state if $w \in L$ or a rejecting state if $w \notin L$ at some time t_0; and for all time $t < t_0$, the output cell is neither in an accepting or rejecting state.

Definition 2.6. The diameter of the path relatively the von Neumann neighborhood and the cell **0** is the minimal radius of the von Neumann ball of center **0** which encompasses the path. L is recognized in real time by $\mathcal{A}$ with p if the output cell **0** enters an accepting or rejecting state at step diameter$(p) - 1$. That is the minimal time that the output cell **0** knows the whole path.

3. The algorithm

While dealing with two-dimensional CA recognizer, specific problems arise when the cells do not know the position of the output cell. To avoid the problems, we suppose that every cell knows the relative distances of its neighbors from each other relatively to the output cell. By instance, with von Neumann neighborhood, a cell in the positive quadrant knows that its closest neighbors relatively to the output cell **0** are its north and west ones, and its farthest neighbors are its south and east ones.

Proposition 3.1. *Providing each cell knows the position of the output cell, any regular language is recognized in real time modulo a constant by a CA equipped with a path, whatever the path may be.*

3.1. The outline

For the sake of simplicity, we may suppose that the path is in the positive quadrant and so that each cell involved in the computation knows that the position of the output cell is in the direction of the northwest. The approach is based on the strategy used by Levialdi [4, 5]

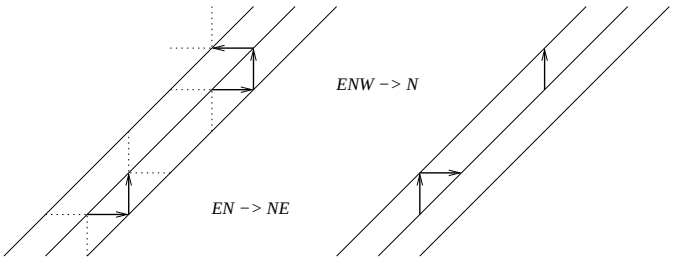

Figure 1: The rewriting rules

which applies local transformations in parallel to shrink object in such a way the diameter of the object decreases of one at each step. Here the CA will shrink the path in applying local rewriting rules to its sequence of moves: **sen** $\rightarrow$ **e** , **swn** $\rightarrow$ **w** , **enw** $\rightarrow$ **n** , **esw** $\rightarrow$ **s** , **sw** $\rightarrow$ **ws** , **en** $\rightarrow$ **ne**; and for the extremities of the path, $\sharp$**n** $\rightarrow \sharp$, $\sharp$**w** $\rightarrow \sharp$, **s**$\sharp \rightarrow \sharp$, **e**$\sharp \rightarrow \sharp$. See Figure 1.
In the same time, the CA will record, above the path, all possible transitions induced locally by the finite automata. Initially, that is, for the cell $\mathbf{c_i}$ which gets the input symbol w_i, the set of transitions $\{(q, \delta(q, w_i)) : q \in Q\}$. Remark that, given the computation which starts

in initial state q_{init}, passes through the states $q_1, \cdots, q_{n-1}$ and ends in state f of the input word w, the tuples $(q_{init}, q_1), (q_1, q_2), \cdots, (q_{n-2}, q_{n-1}), (q_{n-1}, f)$ will be recorded on the cells $\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_{n-1}}, \mathbf{c_n}$ respectively. Together with the shrinkage of the path, the aim will be to update and reduce this sequence of transitions until it will be shorten to the transition (q_{init}, f). Then, according f is or not an accepting state, the result will be transmitted to the output cell.

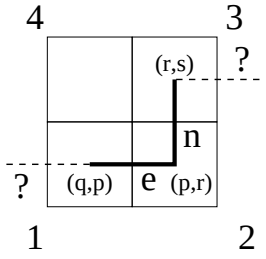

Figure 2: Carrying the rewriting rule $\mathbf{en} \to \mathbf{ne}$

Let us have a look on the rewriting rule $\mathbf{en} \to \mathbf{ne}$ performed at some time t. See Figure 2. Consider $\mathbf{c}$, $\mathbf{c}+\mathbf{e}$, $\mathbf{c}+\mathbf{s}$, $\mathbf{c}+\mathbf{e}+\mathbf{s}$ the four cells involved in the process. At time $t-1$, the path is coded by the three tuples $(?, \mathbf{e})$, $(\mathbf{e}, \mathbf{n})$ and $(\mathbf{n}, ?)$ recorded respectively on the cells $\mathbf{c}+\mathbf{e}$, $\mathbf{c}+\mathbf{e}+\mathbf{s}$ and $\mathbf{c}+\mathbf{s}$. In addition, the cells record sets of transitions. For example, (q, p) belongs to the set of $\mathbf{c}+\mathbf{e}$, (p, r) to $\mathbf{c}+\mathbf{e}+\mathbf{s}$ and (r, s) to $\mathbf{c}+\mathbf{s}$. Now at step t, the four cells have all the required information to perform the rewriting rule $\mathbf{en} \to \mathbf{ne}$, in particular the cell $\mathbf{c}$ can compute the tuple $(\mathbf{n}, \mathbf{e})$ but it needs one more step to compute the transition (p, r). To avoid this slowdown, we will introduce the two following moves: the eastnorth move $\widetilde{\mathbf{en}} = (-1, 1)$ and the southwest $\widetilde{\mathbf{sw}} = (1, -1)$. And every two consecutive moves $\mathbf{e}$ and $\mathbf{n}$ (respectively $\mathbf{s}$ and $\mathbf{w}$) will be coded by only one move $\widetilde{\mathbf{en}}$ (respectively $\widetilde{\mathbf{sw}}$). By the way, the rewriting rules will be modified in this following manner: $\mathbf{n}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{n} \to \mathbf{n}\,\mathbf{n}\,\widetilde{\mathbf{en}}^{\mathbf{k}}$, $\mathbf{e}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{n} \to \widetilde{\mathbf{en}}^{\mathbf{k+1}}$, $\mathbf{s}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{n} \to \widetilde{\mathbf{en}}^{\mathbf{k}}$, $\mathbf{n}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{w} \to \mathbf{n}\,\mathbf{n}\,\widetilde{\mathbf{en}}^{\mathbf{k-1}}$, $\mathbf{e}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{w} \to \widetilde{\mathbf{en}}^{\mathbf{k}}$, $\mathbf{s}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{w} \to \widetilde{\mathbf{en}}^{\mathbf{k-1}}$, $\mathbf{n}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{e} \to \mathbf{n}\,\mathbf{n}\,\widetilde{\mathbf{en}}^{\mathbf{k-1}}\mathbf{e}\,\mathbf{e}$, $\mathbf{e}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{e} \to \widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{e}\,\mathbf{e}$, $\mathbf{s}\,\widetilde{\mathbf{en}}^{\mathbf{k}}\,\mathbf{e} \to \widetilde{\mathbf{en}}^{\mathbf{k-1}}\mathbf{e}\,\mathbf{e}$ and with all the dual rewriting rules ($\mathbf{s}\,\widetilde{\mathbf{sw}}^{\mathbf{k}}\,\mathbf{s} \to \widetilde{\mathbf{sw}}^{\mathbf{k}}\,\mathbf{s}\,\mathbf{s}$, ...) obtained in inverting the direction of the path. Now, at the initialization, we will lose one step to code the path in the right way. But, after, the sequences of transitions will be updated at the same rhythm than the shrinkage of the path.

3.2. The description of the CA

Let be given a FA $\mathcal{F} = (\Sigma, Q, \delta, q_{init}, Q_{\text{accept}})$. The purpose of this section is to describe a CA $\mathcal{A} = (P, S, V, \delta_{\mathcal{A}})$ equipped with a path which recognizes the language $L(\mathcal{F})$.

3.2.1. *The set of states.* First let us define the set of states. It is $S = \Sigma \cup R \times \mathcal{P}(Q^2)$ where $R = \{\mathbf{n}, \mathbf{s}, \mathbf{e}, \mathbf{w}, \widetilde{\mathbf{en}}, \widetilde{\mathbf{sw}}, \sharp\} \times \{\mathbf{n}, \mathbf{s}, \mathbf{e}, \mathbf{w}, \widetilde{\mathbf{en}}, \widetilde{\mathbf{sw}}, \sharp\} \setminus \{(\mathbf{n}, \mathbf{s}), (\mathbf{s}, \mathbf{n}), (\mathbf{e}, \mathbf{w}), (\mathbf{w}, \mathbf{e}), (\mathbf{e}, \mathbf{n}), (\mathbf{s}, \mathbf{w}), (\widetilde{\mathbf{en}}, \mathbf{s}), (\mathbf{n}, \widetilde{\mathbf{sw}}), (\mathbf{w}, \widetilde{\mathbf{en}}), (\widetilde{\mathbf{sw}}, \mathbf{e}), (\widetilde{\mathbf{en}}, \widetilde{\mathbf{sw}}), (\widetilde{\mathbf{sw}}, \widetilde{\mathbf{en}})\}$. Note that the states s belonging to Σ only occur at initial time 0. For states $s = (s_{\text{dir}}, s_{\text{trans}})$ belonging to $R \times \mathcal{P}(Q^2)$, the first component s_{dir} in R codes how the path enters and exits the cell. Because the initial path is simple as well as its rewritings, some consecutive moves never occur. The second component s_{trans} in $\mathcal{P}(Q^2)$ records the set of transitions.

3.2.2. *The preliminary step.* Second let us describe the first step. At initial time 0, the additional layer codes the initial path $p = (\mathbf{c_1}, \cdots, \mathbf{c_n})$ using the set of symbols $P = \{\mathbf{n}, \mathbf{s}, \mathbf{e}, \mathbf{w}, \sharp\} \times \{\mathbf{n}, \mathbf{s}, \mathbf{e}, \mathbf{w}, \sharp\} \setminus \{(\mathbf{n}, \mathbf{s}), (\mathbf{s}, \mathbf{n}), (\mathbf{e}, \mathbf{w}), (\mathbf{w}, \mathbf{e})\}$ and the input words symbols w_i are gotten on the cells $\mathbf{c_i}$. The aim of this preliminary step is to replace every consecutive moves $\mathbf{e}$ and $\mathbf{n}$ (respectively $\mathbf{s}$ and $\mathbf{w}$) of the path by the unique move $\widetilde{\mathbf{en}}$ (respectively $\widetilde{\mathbf{sw}}$). And moreover to record above this path, the transitions induced locally by the FA $\mathcal{F}$. Actually, in one step, all cells have the required information to do the job:

- If the cell $\mathbf{c}$ is outside of the path or the path enters from the East on the cell $\mathbf{c}$ and exits to the North or the path enters from the South and exits to the West (in other words the symbol on the additional layer is $(\sharp, \sharp)$, $(\mathbf{e}, \mathbf{n})$ or $(\mathbf{s}, \mathbf{w})$) then the cell $\mathbf{c}$ enters the quiescent state: $\langle \mathbf{c}, 1\rangle_{\text{dir}} = (\sharp, \sharp)$ and $\langle \mathbf{c}, 1\rangle_{\text{trans}} = \{\}$.
- In case $\mathbf{c}$ is the starting extremity $\mathbf{c_1}$ of the path, if the symbols on the additional layer at position $\mathbf{c}$ and $\mathbf{c} + \mathbf{e}$ (respectively $\mathbf{c}$ and $\mathbf{c} + \mathbf{s}$) are $(\sharp, \mathbf{e})$ and $(\mathbf{e}, \mathbf{n})$ (respectively $(\sharp, \mathbf{s})$ and $(\mathbf{s}, \mathbf{w})$) then $\langle \mathbf{c}, 1\rangle_{\text{dir}} = (\sharp, \widetilde{\mathbf{en}})$ (respectively $(\sharp, \widetilde{\mathbf{sw}})$) and $\langle \mathbf{c}, 1\rangle_{\text{trans}} = \{(q_{init}, \delta(q_{init}, w_1 w_2))\}$. Otherwise $\langle \mathbf{c}, 1\rangle_{\text{dir}}$ takes as value the symbol on the additional layer and $\langle \mathbf{c}, 1\rangle_{\text{trans}} = \{(q_{init}, \delta(q_{init}, w_1))\}$.
- In case $\mathbf{c}$ is the ending extremity $\mathbf{c_n}$ of the path, if the symbols on the additional layer at position $\mathbf{c}$ and $\mathbf{c} + \mathbf{s}$ (respectively $\mathbf{c}$ and $\mathbf{c} + \mathbf{e}$) are $(\mathbf{n}, \sharp)$ and $(\mathbf{e}, \mathbf{n})$ (respectively $(\mathbf{w}, \sharp)$ and $(\mathbf{s}, \mathbf{w})$) then the first component $\langle \mathbf{c}, 1\rangle_{\text{dir}}$ is $(\widetilde{\mathbf{en}}, \sharp)$ (respectively $(\widetilde{\mathbf{sw}}, \sharp)$) otherwise it takes as value the symbol on the additional layer. The second component $\langle \mathbf{c}, 1\rangle_{\text{trans}}$ is $\{(q, f) \in Q^2 : \delta(q, w_n) = f\}$.
- For the remaining cells $\mathbf{c_i}$ on the path, the first component $\langle \mathbf{c_i}, 1\rangle_{\text{dir}}$ is defined in the same way as for the extremities. The second component $\langle \mathbf{c_i}, 1\rangle_{\text{trans}}$ is $\{(q, \delta(q, w_i w_{i+1})) : q \in Q\}$ if the symbols on the additional layer at position $\mathbf{c}$ and $\mathbf{c} + \mathbf{e}$ (respectively $\mathbf{c}$ and $\mathbf{c} + \mathbf{s}$) are $(\sharp, \mathbf{e})$ and $(\mathbf{e}, \mathbf{n})$ (respectively $(\sharp, \mathbf{s})$ and $(\mathbf{s}, \mathbf{w})$) and $\{(q, \delta(q, w_i)) : q \in Q\}$ otherwise.

3.2.3. *The transition function relative to the first component.* Third let us describe the transition function relative to the first component. The different situations required by our CA are depicted in Figure 3. Observe that the north and the west neighbors have no impact in the shrinking process as the output cell is situated in the northwest. Moreover, no direction is given as the rewriting process does not depend on the orientation of the path. For example, the first rule depicts the case where $\langle \mathbf{c}, t\rangle_{\text{dir}} = (\sharp, \sharp)$, $\langle \mathbf{c} + \mathbf{s}, t\rangle_{\text{dir}} = (\mathbf{n}, \widetilde{\mathbf{en}})$ and $\langle \mathbf{c} + \mathbf{e}, t\rangle_{\text{dir}} = (\widetilde{\mathbf{en}}, \mathbf{e})$ as well as the case where $\langle \mathbf{c}, t\rangle_{\text{dir}} = (\sharp, \sharp)$, $\langle \mathbf{c} + \mathbf{e}, t\rangle_{\text{dir}} = (\mathbf{w}, \widetilde{\mathbf{sw}})$ and $\langle \mathbf{c} + \mathbf{s}, t\rangle_{\text{dir}} = (\widetilde{\mathbf{sw}}, \mathbf{s})$. All other possible combinations lead to the quiescent tuple $(\sharp, \sharp)$. Actually some combinations never occur because the initial path is simple as well as its rewritings. We also omit combinations where the extremities are involved. They can be performed in a similar way.

3.2.4. *The transition function relative to the second component.* Finally let us define the updating of the second component of the state. We have to specify how locally the sequence of transitions is either shortened, expanded or shifted according the rewriting of the path. As a state of a FA has at most one successor but possibly several predecessors, the transitions are not symmetrical and their updating depends on the orientation of the path. Below we consider only northeast orientation.

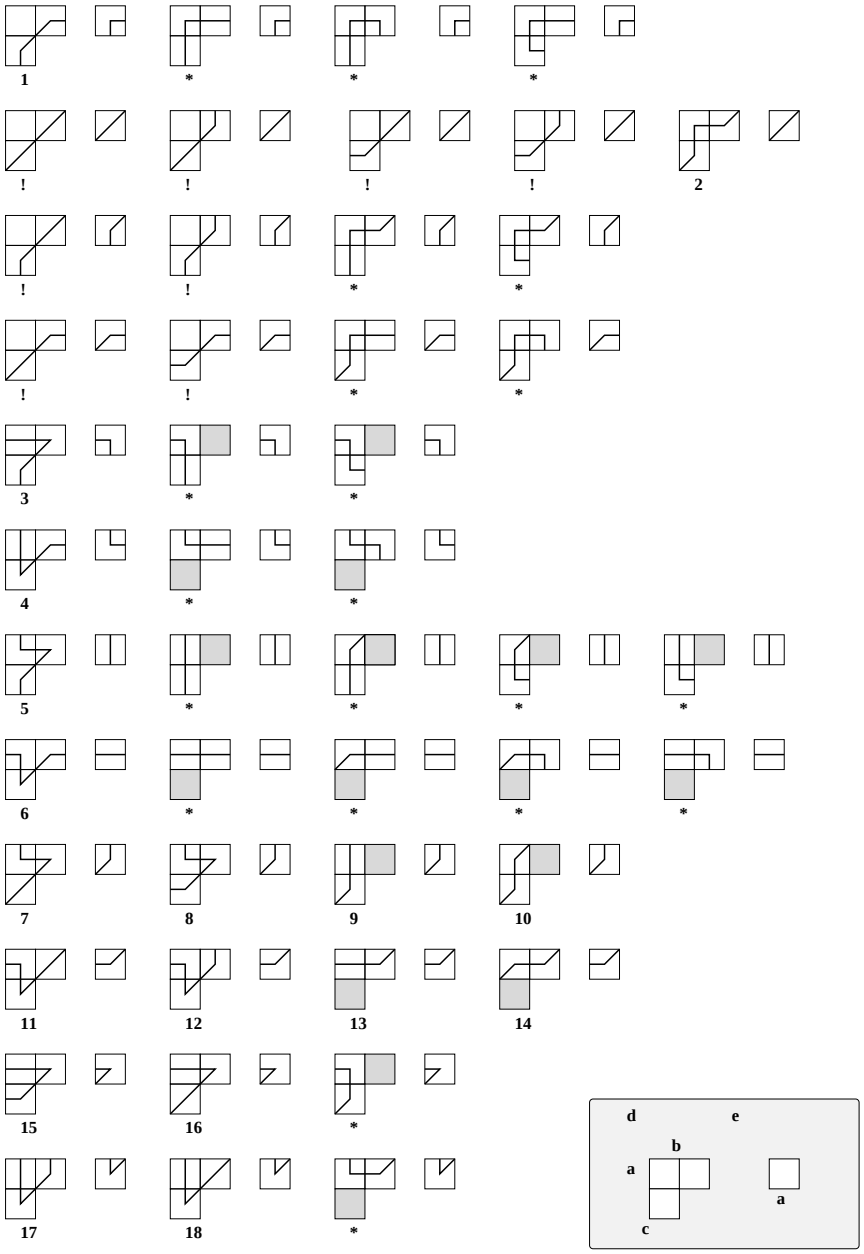

Figure 3: The rules relative to the first component

- For the rules marked with a '$*$' in Figure 3, there is no change: $\langle \mathbf{c}, t+1 \rangle_{\text{trans}} = \langle \mathbf{c}, t \rangle_{\text{trans}}$.
- For the only rule numbered 1, it is an expansion: $\langle \mathbf{c}, t+1 \rangle_{\text{trans}} = \{(q,q) \in Q^2 : \exists p, r$ such that $(p,q) \in \langle \mathbf{c}+\mathbf{s}, t \rangle_{\text{trans}}$ and $(q,r) \in \langle \mathbf{c}+\mathbf{e}, t \rangle_{\text{trans}}\}$.
- For the rules marked with a 'Ⓢ', it is a shift: $\langle \mathbf{c}, t+1 \rangle_{\text{trans}} = \langle \mathbf{c}+\mathbf{e}, t \rangle_{\text{trans}}$.
- For the remaining rules, it is a shrinkage. Precisely $\langle \mathbf{c}, t+1 \rangle_{\text{trans}} = \{(p,q) \in Q^2 : \exists r$ such that $(p,r) \in \langle \mathbf{a}, t \rangle_{\text{trans}}$ and $(r,q) \in \langle \mathbf{b}, t \rangle_{\text{trans}}\}$ with $\mathbf{a} = \mathbf{c}+\mathbf{e}$ and $\mathbf{b} = \mathbf{c}$ for the rules $2, 3, 5, 7, 8, 15$, with $\mathbf{a} = \mathbf{c}$ and $\mathbf{b} = \mathbf{c}+\mathbf{s}$ for the rules $4, 6, 11, 12, 17$,

with $\mathbf{a} = \mathbf{c}+\mathbf{s}$ and $\mathbf{b} = \mathbf{c}$ for the rules $9, 10$, and with $\mathbf{a} = \mathbf{c}$ and $\mathbf{b} = \mathbf{c}+\mathbf{e}$ for the rules $13, 14$. And $\langle \mathbf{c}, t+1\rangle_{\text{trans}} = \{(p,q) \in Q^2 : \exists r, s$ such that $(p,r) \in \langle \mathbf{a}, t\rangle_{\text{trans}}$ and $(r,s) \in \langle \mathbf{b}, t\rangle_{\text{trans}}$ and $(s,q) \in \langle \mathbf{d}, t\rangle_{\text{trans}}\}$ with $\mathbf{a} = \mathbf{c}+\mathbf{s}$, $\mathbf{b} = \mathbf{c}+\mathbf{e}$, $\mathbf{d} = \mathbf{c}$ for the rule 16 and $\mathbf{a} = \mathbf{c}$, $\mathbf{b} = \mathbf{c}+\mathbf{s}$, $\mathbf{d} = \mathbf{c}+\mathbf{e}$ for the rule 18.

An example is given in Figure 4. The tuples of letters along the path represent the computation of the finite automaton. Note that each tuple is just an element of the set of tuples recorded by the second component.

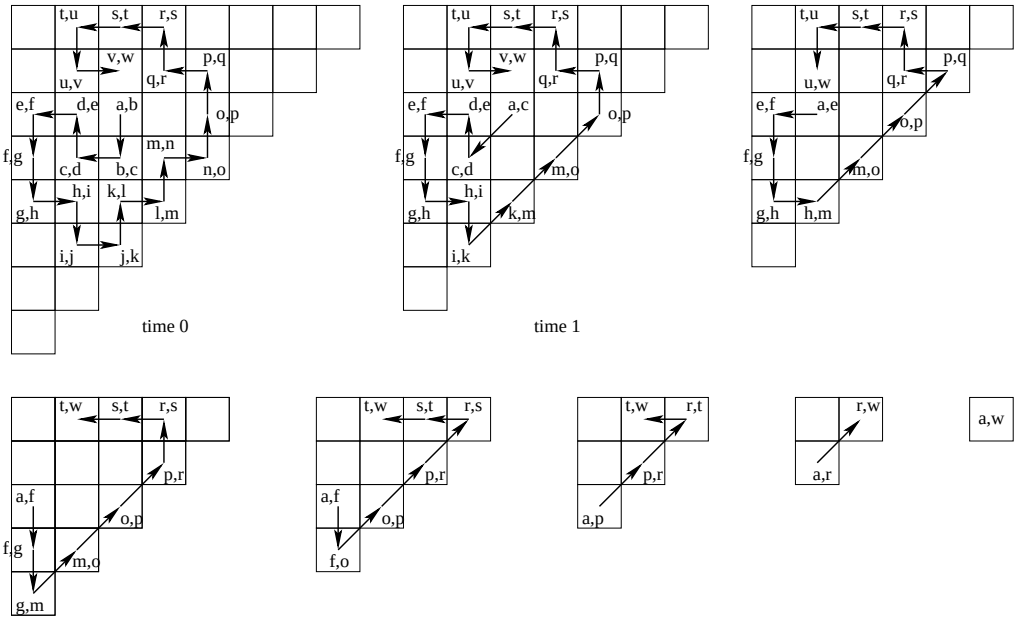

Figure 4: The shrinking process

3.2.5. *Correctness of the algorithm.* According the transition function relative to the first component given in Figure 3, the algorithm rewrites a connected path into a connected path. Moreover, we observe that, if at step t the path does not pass through the south and east borders of the cells $\mathbf{c}+\mathbf{s}$ and $\mathbf{c}+\mathbf{e}$, then at step $t+1$ the path does not pass through the south and east borders of the cell $\mathbf{c}$. Hence, in one step, the algorithm turns a path of diameter d into a path of diameter $d-1$. Finally, remark that the connectivity between the initial state and the final state of the computation of the FA, is preserved by the updating of the second component.

4. Conclusion

The algorithm presented in this paper is based on the strong assumption that each cell knows the relative positions of its neighbors from the output cell. At first glance, it seems unlikely to get rid of this hypothesis. Actually, this problem of orientation is a rather general question concerning CA recognizers in dimension 2 and deserves to be clarified.
We have also assumed that the paths are simple. Clearly the algorithm may be adapted for any path which can go through the same cell more than once but it remains the essential

condition that the path does not cross itself. What happens when we authorize constant bounded crossings of the paths? The algorithm should be modified to avoid unbounded collisions: it will achieve as many rewriting rules as possible (as allowed by the state capacity of the CA) and it will delay the other ones for the next steps. A question is to what extent is the slowdown of the global process linked to these local delays.
Another simplification made was to deal with von Neumann neighborhood instead of Moore neighborhood as it was considered in [2]. So we may wonder whether this shrinking algorithm can be extended to Moore neighborhood.

References

[1] K. Culik. Variations of the firing squad problem and applications. *Information Processing Letters*, 30(3):153–157, February 1989.
[2] M. Delorme and J. Mazoyer. Reconnaissance parallèle des langages rationnels sur automates cellulaires plans. *Theoretical Computer Science*, 281(1–2):251–289, May 2002.
[3] P. C. Fischer. Generation of primes by one-dimensional real-time iterative array. *Journal of the ACM*, 12:388–394, 1965.
[4] S. Levialdi. On shrinking binary picture patterns. *Communications of the ACM*, 15(1):7–10, 1972 .
[5] H. Umeo and G. Mauri. A duality theorem for two connectivity-preserving parallel shrinking transformations. *Future Generation Computer Systems*,18 (7):931–937, August 2002.