Constant Acceleration Theorem for Extended von Neumann Neighbourhoods

Anaël Grandjean(✉)

LIRMM, Université de Montpellier, 161 rue Ada, 34392 Montpellier, France
anael.grandjean@lirmm.fr

Abstract. We study 2-dimensional cellular automata as language recognizers. We are looking for closure properties, similar to the one existing in one dimension. Some results are already known for the most used neighbourhoods, however many problems remain open concerning more general neighbourhoods. In this paper we provide a construction to prove a constant acceleration theorem for extended von Neumann neighbourhoods. We then use this theorem and some classical tools to prove the equivalence of those neighbourhoods, considering the set of languages recognizable in real time.

Introduction

Cellular automata are deterministic dynamical models. Introduced in the 1940s by S. Ulam and J. von Neumann [6] to study self replication in complex systems they were rapidly considered as computation models and language recognizers [4]. Contrary to some other classical computation models that inherently work on words, they can be considered naturally in any dimension (the original cellular automaton studied by Ulam and von Neumann were 2-dimensional) and are therefore particularly well suited to recognize picture languages. Language recognition is performed by encoding the input in an initial configuration and studying the (deterministic) evolution of the automaton from that configuration. Time and space complexities can be defined in the usual way.

One-dimensional cellular automata have been widely studied as language recognizers, and especially concerning real-time and linear time recognition. Some of the most interesting results in this field have been several closure properties as in [1,3,4]. This paper tries to expand two of those properties to 2-dimensional cellular automata.

The first theorem we present is a constant acceleration theorem for some specific set of neighbourhoods. Although such an acceleration is known in one dimension for all neighbourhoods, it was only known in two dimensions for the von Neumann and Moore neighborhoods. This result also extends the constant acceleration theorem for the von Neumann Neighbourhood, based upon a construction from V. Terrier. Althought some other proofs may exists, none is currently published.

M. Cook and T. Neary (Eds.): AUTOMATA 2016, LNCS 9664, pp. 149–158, 2016.
DOI: 10.1007/978-3-319-39300-1_12

The second theorem, which is a consequence of the first one, states the equivalence, with respect to real time recognition, of some neighbourhoods (the ones for which the first theorem is true). Althought in one dimension all complete neighbourhoods are known to be equivalent [1], it has been shown by V. Terrier in [5] that at least two classes of complete neighborhoods exist in two dimensions.

1 Definitions

1.1 Cellular Automata

Definition 1 (Cellular Automaton). *A* cellular automaton *(CA) is a quadruple* $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$ *where*

- $d \in \mathbb{N}$ *is the dimension of the automaton;*
- $\mathcal{Q}$ *is a finite set whose elements are called* states*;*
- $\mathcal{N}$ *is a finite subset of* $\mathbb{Z}^d$ *called* neighbourhood *of the automaton;*
- $\delta : \mathcal{Q}^{\mathcal{N}} \to \mathcal{Q}$ *is the* local transition function *of the automaton.*

Definition 2 (Configuration). *A* d-dimensional configuration $\mathfrak{C}$ *over the set of states* $\mathcal{Q}$ *is a mapping from* $\mathbb{Z}^d$ *to* $\mathcal{Q}$*.*

The elements of $\mathbb{Z}^d$ *will be referred to as* cells *and the set of all* d*-dimensional configurations over* $\mathcal{Q}$ *will be denoted as* $\mathrm{Conf}_d(\mathcal{Q})$*.*

Given a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, a configuration $\mathfrak{C} \in \mathrm{Conf}_d(Q)$ and a cell $c \in \mathbb{Z}^d$, we denote by $\mathcal{N}_{\mathfrak{C}}(c)$ the neighbourhood of c in $\mathfrak{C}$:

$$\mathcal{N}_{\mathfrak{C}}(c) : \begin{cases} \mathcal{N} \to \mathcal{Q} \\ n \mapsto \mathfrak{C}(c+n) \end{cases}$$

From the local transition function δ of a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, we can define the *global transition function of the automaton* $\Delta : \mathrm{Conf}_d(\mathcal{Q}) \to \mathrm{Conf}_d(\mathcal{Q})$ obtained by applying the local rule on all cells:

$$\Delta(\mathfrak{C}) = \begin{cases} \mathbb{Z}^d \to \mathcal{Q} \\ c \mapsto \delta(\mathcal{N}_{\mathfrak{C}}(c)) \end{cases}$$

The action of the global transition rule makes $\mathcal{A}$ a dynamical system over the set $\mathrm{Conf}_d(\mathcal{Q})$. Because of this dynamics, in the following we will identify the CA $\mathcal{A}$ with its global rule so that $\mathcal{A}(\mathfrak{C})$ is the image of a configuration $\mathfrak{C}$ by the action of the CA $\mathcal{A}$. More generally $\mathcal{A}^t(\mathfrak{C})$ is the configuration resulting from applying t times the global rule of the automaton from the initial configuration $\mathfrak{C}$.

Definition 3 (Von Neumann and Moore Neighbourhoods). *In* d *dimensions, the most commonly considered neighbourhoods are the von Neumann neighbourhood* $\mathcal{N}_{\mathrm{vN}} = \{c \in \mathbb{Z}^d,\ ||c||_1 \leq 1\}$ *and the Moore neighbourhood* $\mathcal{N}_{\mathrm{M}} = \{c \in \mathbb{Z}^d, ||c||_\infty \leq 1\}$*. Figure 1 illustrates these two neighbourhoods in 2 dimensions.*

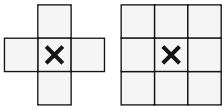

Fig. 1. The von Neumann (left) and Moore (right) neighbourhoods in 2 dimensions.

Definition 4 (a-b-Neighbourhood). *We denote by* a-b-Neighbourhood *(shortly* $N_{a,b}$*), the following two-dimensional neighbourhood:*

$$N_{a,b} = \{(x, y) \in \mathbb{Z}^2 \mid b|x| + a|y| \leq ab\}$$

Note that such a neighbourhood is convex and symmetric with respect to the origin. For completeness reasons we also require a and b to be strictly positive. Furthermore, the neighbourhood $N_{1,1}$ is exactly the von Neumann Neighbourhood. Some examples are depicted in Fig. 2.

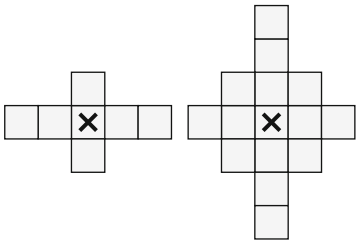

Fig. 2. $N_{2,1}$ (left) and $N_{2,3}$ (right)

1.2 Picture Recognition

From now on we will only consider 2-dimensional cellular automata (2DCA), and the set of cells will always be $\mathbb{Z}^2$.

Definition 5 (Picture). *For* $n, m \in \mathbb{N}$ *and* Σ *a finite alphabet, an* (n, m)-picture *(picture of width* n *and height* m*) over* Σ *is a mapping*

$$p : [\![0, n-1]\!] \times [\![0, m-1]\!] \to \Sigma$$

$\Sigma^{n,m}$ *denotes the set of all* (n, m)*-pictures over* Σ *and* $\Sigma^{*,*} = \bigcup_{n,m \in \mathbb{N}} \Sigma^{n,m}$ *the set of all pictures over* Σ*. A* picture language *over* Σ *is a set of pictures over* Σ*.*

Definition 6 (Picture Configuration). *Given an* (n, m)*-picture* p *over* Σ*, we define the* picture configuration *associated to* p *with quiescent state* $q_0 \notin \Sigma$ *as*

$$\mathfrak{C}_{p,q_0} : \begin{cases} \mathbb{Z}^2 \to \Sigma \cup \{q_0\} \\ x, y \mapsto \begin{cases} p(x, y) & \text{if } (x, y) \in [\![0, n-1]\!] \times [\![0, m-1]\!] \\ q_0 & \text{otherwise} \end{cases} \end{cases}$$

Definition 7 (Picture Recognizer). *Given a picture language L over an alphabet Σ, we say that a 2DCA $\mathcal{A} = (2, \mathcal{Q}, \mathcal{N}, \delta)$ such that $\Sigma \subseteq \mathcal{Q}$ recognizes L with quiescent state $q_0 \in \mathcal{Q} \setminus \Sigma$ and accepting states $\mathcal{Q}_a \subseteq \mathcal{Q}$ in time $\tau : \mathbb{N}^2 \to \mathbb{N}$ if, for any picture p (of size $n \times m$), starting from the picture configuration $\mathfrak{C}_{p,q_0}$ at time 0, the origin cell of the automaton is in an accepting state at time $\tau(n,m)$ if and only if $p \in L$. Formally,*

$$\forall n, m \in \mathbb{N}, \forall p \in \Sigma^{n,m}, \quad \mathcal{A}^{\tau(n,m)}(\mathfrak{C}_{p,q_0})(0,0) \in \mathcal{Q}_a \Leftrightarrow p \in L$$

We then say that the language L can be recognized in time $\tau(n,m)$ with neighbourhood N.

Since cellular automata work with a finite neighbourhood, the state of the origin cell at time t (after t actions of the global rule) only depends on the initial states on the cells in $\mathcal{N}^t$, where $\mathcal{N}^0 = \{0\}$ and for all n, $\mathcal{N}^{n+1} = \{x + y,\ x \in \mathcal{N}^n, y \in \mathcal{N}\}$. The real time function is informally defined as the smallest time such that the state of the origin may depend on all letters of the input:

Definition 8 (Real Time). *Given a neighbourhood $\mathcal{N} \subset \mathbb{Z}^d$ in d dimensions, the real time function $\tau_{\mathcal{N}} : \mathbb{N}^d \to \mathbb{N}$ associated to $\mathcal{N}$ is defined as*

$$\tau_{\mathcal{N}}(n_1, n_2, \ldots, n_d) = \min\{t, [\![0, n_1 - 1]\!] \times [\![0, n_2 - 1]\!] \times \ldots \times [\![0, n_d - 1]\!] \subseteq \mathcal{N}^t\}$$

When considering the specific case of the 2-dimensional von Neumann neighbourhood, the real time is defined by $\tau_{\mathcal{N}_{\text{vN}}}(n,m) = n + m - 2$. There is however a well known constant speed-up result:

Proposition 1 (folklore). *For any $k \in \mathbb{N}$, any language that can be recognized in time $(\tau_{\mathcal{N}_{\text{vN}}} + k)$ by a 2DCA working on the von Neumann neighbourhood can also be recognized in real time by a 2DCA working on the von Neumann neighbourhood.*

So, it will be enough to prove that a language is recognized in time $(n,m) \mapsto n + m + k$ for some constant k to prove that it is recognized in real time.

2 Main Result

Theorem 1 (constant acceleration). *For any a, b and k positive integers, any language that can be recognized in time $(\tau_{N_{a,b}} + k)$ by a 2DCA working on $N_{a,b}$ can also be recognized in real time by a 2DCA working on the same neighbourhood.*

Proof. To prove the theorem one only needs to be able to accelerate one step of the calculation (that is, recognize in real time any language recognized in real time plus one). We then only have to repeat the process a finite number of times.

As in the one dimensional case, the idea is to make each cell "guess" the state of some further cells, so that at real time, the origin cell guesses are correct, and allow it to perform one more step of computation.

Fix some language L and some automaton A recognizing this language in real time plus one step. This automaton works on the a-b-Neighbourhood $N_{a,b}$. We will now construct an automaton A' with the same neighbourhood, recognizing L in real time.

The potential initial configurations of both automata are the same, from now on we fix one initial configuration and explain what happens on the run starting from this configuration in A', depending on what happens in the run starting from this same configuration in A.

Let us introduce some notations: $A(c,t)$ represents the state of the cell c at time t in the original automaton A. We denote as G_{all} the following set:

$$G_{all} = \{(xa, yb) \in \mathbb{Z}^2 \| 1 < x + y \leq 2; 0 \leq x; 0 \leq y; x \leq y + 1; y \leq x + 1\}$$

This is a subset of the north east quarter of N^2, depicted in green in Fig. 3. Similarly we define G_{left} and G_{bottom} as follows, depicted in light blue in Fig. 3:

$$G_{left} = \{(xa, yb) \in \mathbb{Z}^2 \| 1 < x + y \leq 2; 0 \leq x; 0 \leq y; x \leq y + 1; y \geq x + 1\}$$

$$G_{bottom} = \{(xa, yb) \in \mathbb{Z}^2 \| 1 < x + y \leq 2; 0 \leq x; 0 \leq y; x \geq y + 1; y \leq x + 1\}$$

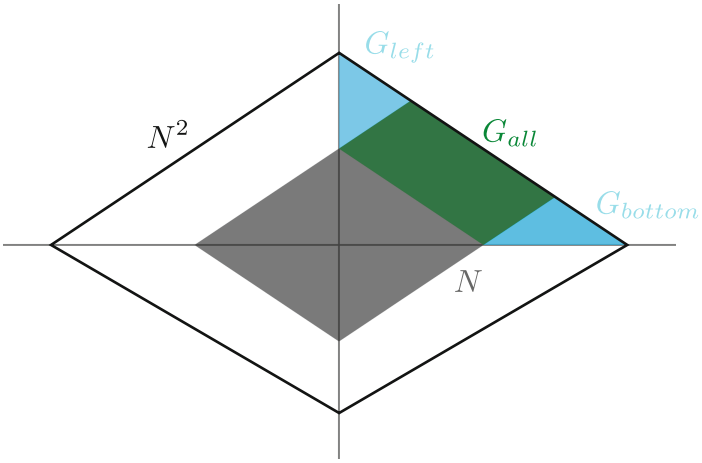

Fig. 3. A partition of N^2 (Color figure online)

Each cell c of the new automaton A' will "contain" many states of cells of A. More formally the state set of A' is a power of the state set of A. A cell c of A' contains, at time $t \geq 1$, the following informations:

- $A(c,t)$
- $A(c', t-1)$ for every $c' \in N(c)$.
- $g(c, c', t-1)$ for every $c' \in G_{all}(c)$.

- $g(c, c', t-1)$ for every $c' \in G_{bottom}(c)$ if c is on the bottom border of the input word.
- $g(c, c', t-1)$ for every $c' \in G_{left}(c)$ if c is on the left border of the imput word.

We call $E(c)$ (extended neighbourhood) the set of cells c' such that c holds either $A(c', t)$ or $g(c, c', t)$. Remark that $E(0)$ is exactly the north east quarter of N^2.

The state $g(c, c', t)$ is some sort of guess, made by the cell c, of what could be $A(c', t)$. For every cells c and c', $g(c, c', 0) = \#$, the quescient state of A. The update rule of $g(c)$ will depend on the available information for the cell c. The new state $g(c, c', t+1)$, is the result of applying the local rule of A on the neighbours of c', using the state of A when it is available to c, and a guess of another cell otherwise.

To better understand this update rule, we need to focus on what information is available for each cell of A' during the computation. A cell c can "see" every information contained its neighbouring cells. This way, for example, at time t, a cell c have access to all $A(c', t-1)$ for $c' \in N^2(c)$, as each cell of its neighbourhood contain this information for each cell of its own neighbourhood. This way it is easy to update every state of A, directly using the local rule of A.

Remark that one cell can see several times the same information, as it is contained in more than one cell of its neighbourhood. As the information about the states of $A(c, t)$ results from a direct simulation, all its occurences are coherent. On the contrary, guesses about one cell state can depend on which cell is making the guess. To avoid incoherence issues, a cell c will only use the information contained in the guesses of its leftmost and uppermost neighbours, and completely ignore all other guesses (including its own previous guesses).

Some quick math calculations shows that the neighbourhood of each cell in $E(c)$ is indeed contained in $N^2(c) \cup E(c + (a, 0)) \cup E(c + (0, b))$, as depicted in Fig. 4. A cell whose state is # formally stays in this state, but is "read" as if its extended neighbourhood was filled with #.

This equation is not true for the cells on the left border and on the bottom border, those cells needs to use some guesses about cells outside the computation. However those cells will always remain in the quescient state, therefore no information is actually missing.

We will say that a cell is correct if all the $g(c, c', t)$ in its extended neighbourhood are equal to $A(c', t)$. After the first time step, all the cells of the upmost row and of the rightmost column are correct. Moreover all the cells with the top right corner cell in its neighbourhood are correct. As all the cells outside the computation also correct, we can note that each cell which is above or on the right of a correct cell is also correct.

At time $t+1$ a cell c is correct if both $c + a$ and $c + b$ are correct. This is also true for the cells on the borders, using the fact that every cell outside the border is in state #.

Because of the shape of the set of correct cells at time 1, it is easy to see that a time t, the set of correct cells will contain $N^{-t}((n, m))$ where n and m are respectively the length and height of the input.

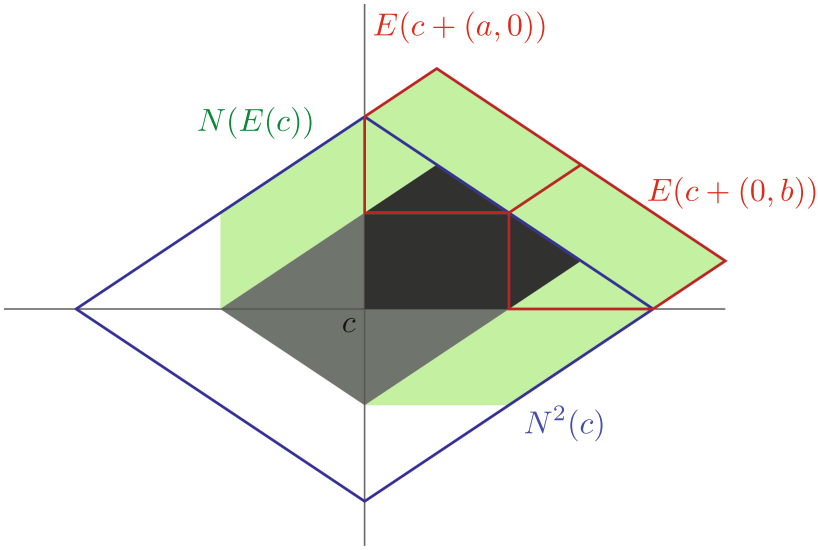

Fig. 4. Available and needed information for updating cells

Because of the shape of the neighbourhood, this cell is the farthest from the origin. Then, the real time is equal to the minimal t such that the origin cell is in $N^{-t}(n,m)$. Thus, at real time, denoted $\tau(n,m)$, the origin cell is correct, and therefore knows the state of every cell of its extended neighbourhood at time $\tau-1$, in the automaton A. The extended neighbourhood of the origin consists of all the cells in $N^2(0)$ which are not in state #. This information is enough to compute the state of all cells in $N((0,0))$ at time τ in A. Then it is engouh information to compute the state of the origin cell at time $\tau+1$ in A.

The accepting states of A' are all the "configurations" of the extended neighbourhood of the origin cell which lead to an acceptation in A at time $\tau+1$.

Theorem 2 (Equivalence of the a-b-Neighbourhoods). *For any a,b,c,d positive integers, any language that can be recognized in real time by an automaton with $N_{a,b}$ can also be recognized in real time by an automaton with $N_{c,d}$.*

Proof. The proof of this theorem is based upon the two following lemmas:

Lemma 1. *For any a, b and k positive integers, the sets of languages recognized in real time with $N_{a,b}$ and with $N^k_{a,b}$ are the same. We say that $N_{a,b}$* and *$N^k_{a,b}$ are* equivalent *with respect to the real time.*

Lemma 2. *For any a, b and k positive integers, any language recognized in real time with $N_{ka,b}$ (symmetrically $N_{a,kb}$) can also be recognized in real time with $N_{a,b}$. We say that $N_{a,b}$ is* more powerful *than $N_{ka,b}$.*

The first lemma is well known for one dimensional cellular automata, and is an easy corollary of the constant acceleration theorem. First note that the real time function for N and the one for N^k are very similar. Indeed $\tau_{N^k} = \lceil \frac{\tau_N}{k} \rceil$.

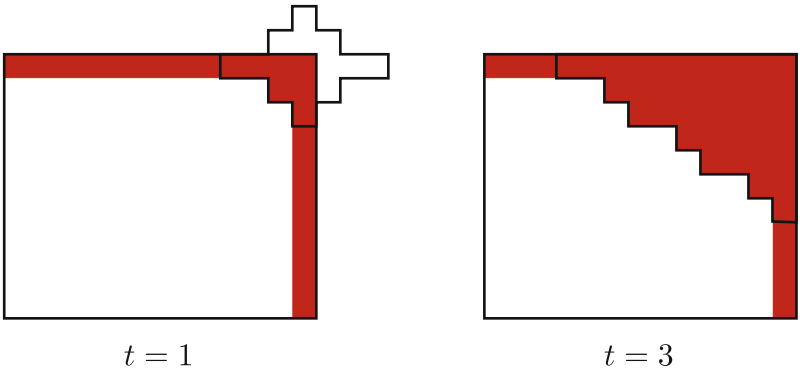

Fig. 5. Correct cells at the first steps

As an automaton A' with neighbourhood N^k can simulate k steps of computation of an automaton A with neighbourhood N in a single step, in real time it can simulate at least τ_N steps of A, proving one inclusion (Fig. 5).

The other inclusion needs the constant acceleration theorem. Indeed, an automaton A' with neighbourhood N needs k steps to simulate one step of an automaton A with the neighbourhood N^k. Thus it needs at most $\tau_N + k$ steps to compute τ_{n^k} steps of A. Thanks to the first theorem of this paper, we can build another automaton with neighbourhood N recognizing the same language in real time, completing the proof.

In order to prove the second lemma, we will have to perform a compression of the input. Let A be an automaton recognizing L with neighbourhood $N_{ka,b}$ in real time. Once again the two neighbourhoods we consider have very similar real time:

$$\tau_{N_{ka,b}}(n,m) = \lceil \frac{n}{ka} \rceil + \lceil \frac{m}{b} \rceil$$

$$\tau_{N_{a,b}}(n,m) = \lceil \frac{n}{a} \rceil + \lceil \frac{m}{b} \rceil$$

Now consider the following neighbourhood:

$$M = \{(x,0) | -ka \leq x \leq ka\} \cup \{(0,y) | -b \leq y \leq b\}$$

M have exactly the same convex hull as $N_{ka,b}$, but is not convex. However thanks to Delacourt and Poupet [2] we know that there is an automaton A' with neighbourhood M which recognizes L in time at most $\tau_{N_{ka,b}} + c$ for some constant c.

We will now build an automaton B which simulates A'. Each cell of B will be able to store up to k states of A'. First the automaton will perform a compression of the input, line by line, by a factor k. This takes $\lceil \frac{((k-1)n}{ka} \rceil$ time steps.

After the compression each cell of B contains k cells of A', and can simulate the computation of A' for each of those cells without losing any time. In Fig. 6 the automaton B is represented after a compression by factor 3. Each blue

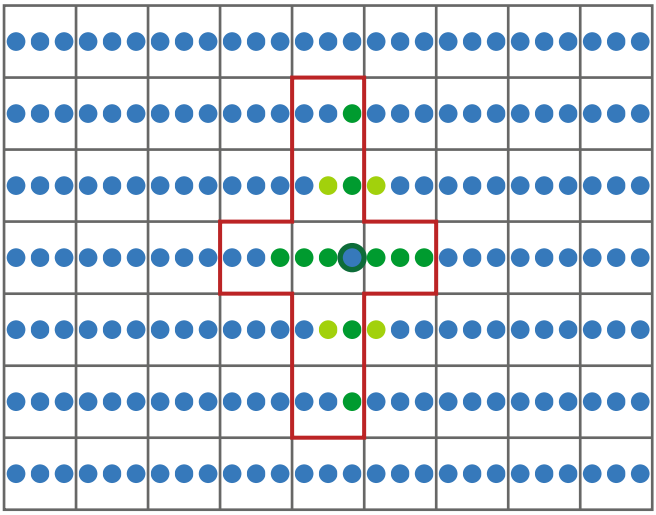

Fig. 6. Available information after compression (Color figure online)

dot in a cell correspond to a cell of A'. Here, $N_{1,2}$ is the neighbourhood of B, depicted in red. $N_{3,2}$ is the neighbourhood of A. The green dots represents the neighbourhood of the blue dot circled in green. The four light green dots are the one which are in $N_{3,2}$ but not in M with respect to the circled dot.

By doing this simulation, at time $\tau_{N_{a,b}} + c$ the automaton B have simulated $\tau_M + c$ steps of A', recognizing language L. Thanks to the constant acceleration theorem, there is an automaton B' which can also recognize L in real time with neighbourhood $N_{a,b}$.

Now lets go back to the proof of our theorem. Consider four integers a, b, c and d, N the a-b-Neighbourhood, and M the c-d-Neighbourhood. We denote by M' the abc-abd-Neighbourhood and N' the abc-b-neighbourhood. By the second lemma, we know that $N_{a,b}$ is more powerful than $N_{abc,b}$. By applying this lemma again we have that $N_{abc,b}$ is more powerful than $N_{abc,abd}$. Therefore $N_{a,b}$ is more powerful than $N_{abc,abd}$. Remark that $N^{ab}_{c,d} = N_{abc,abd}$. Thanks to the first lemma we know that $N_{c,d}$ and $N_{abc,abd}$ are equivalent with respect to real time recognition. Thus $N_{a,b}$ is more powerful than $N_{c,d}$.

With similar ideas we can prove that $N_{c,d}$ is more powerful than $N_{a,b}$, proving the announced result.

References

1. Poupet, V.: Cellular automata: real-time equivalence between one-dimensional neighborhoods. Theor. Comput. Syst. **40**(4), 409–421 (2007)
2. Delacourt, M., Poupet, V.: Real time language recognition on 2D cellular automata: dealing with non-convex neighborhoods. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 298–309. Springer, Heidelberg (2007)

3. Mazoyer, J., Reimen, N.: A linear speed-up theorem for cellular automata. Theor. Comput. Sci. **101**, 59–98 (1991)
4. Smith III, A.R.: Real-time language recognition by one-dimensional cellular automata. J. ACM **6**, 233–253 (1972)
5. Terrier, V.: Two-dimensional cellular automata recognizer. Theor. Comput. Sci. **218**(2), 325–346 (1999)
6. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)