Cellular Automata: Real-Time Equivalence Between One-Dimensional Neighborhoods

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. It is well known that one-dimensional cellular automata working on the usual neighborhood are Turing complete, and many acceleration theorems are known. However very little is known about the other neighborhoods. In this article, we prove that every one-dimensional neighborhood that is sufficient to recognize every Turing language is equivalent (in terms of real-time recognition) either to the usual neighborhood $\{-1, 0, 1\}$ or to the one-way neighborhood $\{0, 1\}$.
Keywords: Cellular automata, neighborhoods, language recognition, real-time.

1 Introduction

Cellular automata (CA) are a computation model that is simple at microscopic scale (local transitions) but can have very complex behaviours at macroscopic scale (global transitions). As a complex systems, it is natural to wonder what their compuational capabilities are. It is known that they can simulate any Turing Machine [15,10,1] (Turing universality) and that there exist "universal" CA that can simulate the evolution of any other [8] (intrinsic universality).

Once their computational power is known, we investigate the complexity of such computations (in time and memory). Many famous algorithms [6,5] have shown that the massively parallel structure of the CA enables complex computations to be realized in a very short time. However, because of the local communication between cells, it is possible to show that information cannot be transmitted faster than some "maximal speed".

In this article, we will work on the problem of language recognition by CA, and more specifically the recognition of languages in "real time", which is the fastest possible time according to the "maximal speed" restriction. Many results are already known on this topic concerning one-dimensional CA working on the standard neighborhood [11,7,4,9,12,13], but very little is known about CA working on different neighborhoods.

Concerning Turing machines, it is easy to show that any computation can be accelerated by a constant time (as long as the time stays greater than the minimal time necessary to read the entry). As for cellular automata, C. Choffrut and K. Čulik in [2] first show that with the neighborhood $\{-1, 0, 1\}$, any computation on a CA can be accelerated by a constant time. In [9], J. Mazoyer and N. Reimen show that for any positive integers r and q, the neighborhoods $[-r, r]$ and $[-q, q]$ are time wise equivalent. Moreover, S. Cole [3] showed that on every neighborhood “complete enough” (such that every cell can, after some finite time, be affected by the state of any other) it is possible to simulate the behaviour of any CA working on any other neighborhood. This last result gives a first computational equivalence between neighborhoods.

Here, a strong “real-time equivalence” between neighborhoods will be proved that shows that the languages recognized in real-time by CA working on a given neighborhood “complete enough” (similarly to Cole’s definition) are either exactly the same as the ones recognized in real-time by CA working on the usual neighborhood or exactly the ones recognized in real-time by CA working on the one-way neighborhood (one-way cellular automata).

2 Preliminary Study: Neighborhoods Growth

In all this section, a *neighborhood* will be a finite subset of $\mathbb{Z}$. We will here study an algebraic property of neighborhoods that is independent of the notion of cellular automaton.

Given a neighborhood V, we will use the notations $V^0 = \{0\}$ and for all $k \in \mathbb{N}$, $V^{k+1} = \{x + y | x \in V, y \in V^k\}$. Moreover, we will always consider that the neighborhoods contain 0.

Definition 21 *Let V be a neighborhood. V is*

- r-complete *if* $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}, n \in V^k$*;*
- complete *if* $\forall n \in \mathbb{Z}, \exists k \in \mathbb{N}, n \in V^k$*;*
- r-incomplete *if it is* r-complete *but not* complete*.*

Proposition 21 *A neighborhood V is r-incomplete if and only if it contains 1 and has no negative element. It is complete if and only if its non-zero elements are prime altogether (their gcd is 1) and has both a positive and a negative element.*

Proof. It is obvious that a neighborhood that contains 1 and has no negative element is r-incomplete. Reciprocally, if it doesn’t contain 1 and has no negative element, then all possible non-zero sums of its elements are greater or equal to its smallest non-zero element, and 1 cannot be obtained. If it has a negative element $(-x_n)$, then either 1 can be obtained, and therefore the neighborhood is complete ($\forall x, x = p.(-x_n) + q.1$ for some p and q in $\mathbb{N}$) or 1 cannot be obtained and V is incomplete.

To prove the second equivalence, consider $V = \{0, v_1, \ldots, v_s\}$.
If $\gcd(v_1, \ldots, v_s) = 1$ then by Bezout’s theorem,

$$\exists(\mu_1, \ldots, \mu_s) \in \mathbb{Z}^s, \quad \sum_{i \in [\![1,n]\!]} \mu_i v_i = 1$$

And so, as seen before, we have both 1 and a negative element so the neighborhood is complete. Reciprocally, if the neighborhood doesn't have any positive element (or negative) then it can obviously not be complete, and if all elements are not prime altogether, then they are all multiples of an integer $k > 1$ and only multiples of k can be obtained and V is not complete. $\square$

Remark. We will see later that the r-complete neighborhoods are the neighborhoods on which a CA can correctly recognize a language.

From now on, if V is an r-complete neighborhood, we will use the following notations

$$\begin{aligned} x_p &= \max V \\ -x_n &= \min V \\ t_c &= \min\{t \in \mathbb{N} | [\![-x_n, x_p]\!] \subseteq V^{t+1}\} \end{aligned}$$

Remark. Proposition 21 ensures that t_c is correctly defined.

Proposition 22 *Let V be an r-complete neighborhood. For all $t \in \mathbb{N}$, we have*

$$[\![-tx_n, tx_p]\!] \subseteq V^{t+t_c} \subseteq [\![-(t+t_c)x_n, (t+t_c)x_p]\!]$$

(see fig. 1)

Proof. The rightmost inclusion is trivial. As for the leftmost one, we prove it inductively. For $t = 0$ and $t = 1$, the property is obvious. For $t \geq 1$, we assume that $[\![-tx_n, tx_p]\!] \subseteq V^{t+t_c}$. So

$$[\![-(t+1)x_n, (t+1)x_p]\!] = [\![-tx_n, tx_p]\!] + \{-x_n, 0, x_p\} \subseteq V^{t+t_c} + V = V^{t+t_c+1}$$

$\square$

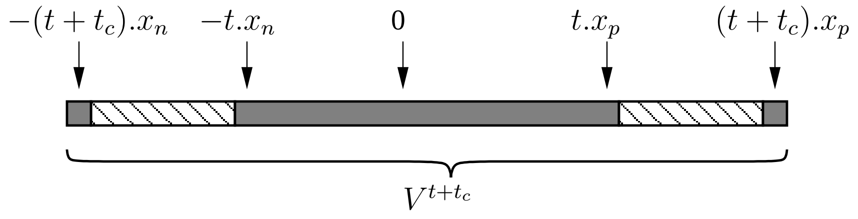

Fig. 1. The general form of $V^{(t+t_c)}$

Definition 22 *Let V be an r-complete neighborhood. We will call* r-shadow *of V at time $(t+t_c)$ the set*

$$S^+_{t+t_c}(V) = \left(V^{t+t_c} \cap [\![tx_p, (t+t_c)x_p]\!]\right) - tx_p$$

Similarly, we'll call l-shadow *of V at time $(t + t_c)$ the set*

$$S^-_{t+t_c}(V) = \left(V^{t+t_c} \cap [\![-(t+t_c)x_n, -tx_n]\!]\right) + tx_n$$

Remark. According to the proposition 22,

$$V^{t+t_c} = \left[S^-_{t+t_c}(V) - tx_n\right] \cup [\![-tx_n, tx_p]\!] \cup \left[S^+_{t+t_c}(V) + tx_p\right]$$

Proposition 23 *Let V be an r-complete neighborhood. The sequences*

$$\left(S^+_{t+t_c}(V)\right)_{t\in\mathbb{N}} \text{ and } \left(S^-_{t+t_c}(V)\right)_{t\in\mathbb{N}}$$

are ultimately constant.

Proof. Let $x \in S^+_{t+t_c}(V)$, then $(x + tx_p) \in V^{t+t_c}$ and so $(x + (t+1)x_p) \in V^{t+t_c+1}$ and finally $x \in S^+_{t+1+t_c}(V)$. This proves that $\left(S^+_{t+t_c}(V)\right)_{t\in\mathbb{N}}$ is an increasing sequence (where the considered order is the inclusion). Moreover, this sequence's elements are all in the subsets of $[\![0, t_cx_p]\!]$, which is a finite set. This implies that the sequence is ultimately constant. The proof is similar for $\left(S^-_{t+t_c}(V)\right)_{t\in\mathbb{N}}$. □

Definition 23 *Let V be an r-complete neighborhood. The* stabilization time *t_s of V is the smallest integer from which the sequences $\left(S^+_{t+t_c}(V)\right)_{t\in\mathbb{N}}$ and $\left(S^-_{t+t_c}(V)\right)_{t\in\mathbb{N}}$ are constant.*

We also define $S^+(V) = S^+_{t_c+t_s}$, $S^-(V) = S^-_{t_c+t_s}$ and $x_0 = \min\{x \in \mathbb{N} | x \notin S^+(V)\}$.

Proposition 23 ensures that t_s is correctly defined for all r-complete neighborhoods. We have proven the following theorem

Theorem 21 *For all r-complete neighborhood V, there exists an integer t_s, and two sets $S^-(V) \subseteq [\![-t_cx_n, 0]\!]$ and $S^+(V) \subseteq [\![0, t_cx_p]\!]$ such that for all $t \geq t_s$,*

$$V^{t+t_c} = \left(S^-(V) - tx_n\right) \cup [\![-tx_n, tx_p]\!] \cup \left(S^+(V) + tx_p\right)$$

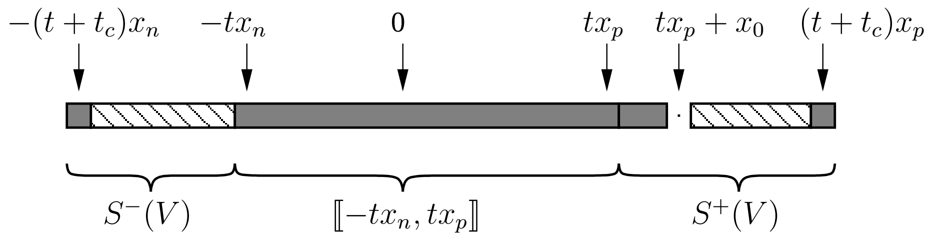

Fig. 2. The general form of $V^{(t+t_c)}$ for $t \geq t_s$

3 Language Recognition by Cellular Automata

3.1 Cellular Automata

Definition 31 *A* cellular automaton *(CA) is a triple $\mathcal{A} = (\mathcal{Q}, V, f)$ where*

- *$\mathcal{Q}$ is a finite set called* set of states *containing a special* quiescent *state #;*
- *$V = \{v_1, \dots, v_{|V|}\} \subseteq \mathbb{Z}$ is a finite set called* neighborhood *that contains 0.*
- *$f : \mathcal{Q}^{|V|} \to \mathcal{Q}$ is the* transition function*. We have $f(\#, \dots, \#) = \#$.*

For a given automaton $\mathcal{A}$, we call *configuration* of $\mathcal{A}$ any function $\mathfrak{C}$ from $\mathbb{Z}$ into $\mathcal{Q}$. The set of all configurations is therefore $\mathcal{Q}^{\mathbb{Z}}$. From the local function f we can define a global function F

$$\begin{aligned} F : \mathcal{Q}^{\mathbb{Z}} &\to \mathcal{Q}^{\mathbb{Z}} \\ \mathfrak{C} &\mapsto \mathfrak{C}' \quad | \quad \forall x \in \mathbb{Z}, \mathfrak{C}'(x) = f(\mathfrak{C}(x+v_1), \dots, \mathfrak{C}(x+v_{|V|})) \end{aligned}$$

Elements of $\mathbb{Z}$ are called *cells*. Given a configuration $\mathfrak{C}$, we'll say that a cell c is in state q if $\mathfrak{C}(c) = q$.

If at time $t \in \mathbb{N}$ the CA is in a configuration $\mathfrak{C}$, we'll consider that at time $(t+1)$ it is in the configuration $F(\mathfrak{C})$. This enables to define the *evolution* of a CA from a configuration. This evolution is completely determined by $\mathfrak{C}$.

3.2 Language Recognition

Definition 32 *We consider a CA $\mathcal{A} = (\mathcal{Q}, V, f)$ and a set $\mathcal{Q}_{acc} \subseteq \mathcal{Q}$ of* accepting states*. Let $w = w_0 w_1 \dots w_{l-1}$ be a word on a finite alphabet $A \subseteq \mathcal{Q}$. We define the configuration $\mathfrak{C}_w$ as follows.*

$$\begin{aligned} \mathfrak{C}_w : \quad & \mathbb{Z} \to \mathcal{Q} \\ & \begin{cases} x \mapsto w_x & \text{if } 0 \leq x < l \\ x \mapsto \# & \text{otherwise} \end{cases} \end{aligned}$$

We'll say that the CA $\mathcal{A}$ recognizes *the word w with accepting states $\mathcal{Q}_{acc}$ in time t_w if, starting from the configuration $\mathfrak{C}_w$ at time 0, the cell 0 is in a state in $\mathcal{Q}_{acc}$ at time t_w.*

Definition 33 *Let $\mathcal{A} = (\mathcal{Q}, V, f)$ be a CA and $L \subseteq A^*$ a language on the alphabet $A \subseteq \mathcal{Q}$. For a given function T from $\mathbb{N}$ into $\mathbb{N}$, we'll say that the language L is* recognized by $\mathcal{A}$ in time T *if there exists a set $\mathcal{Q}_{acc} \subseteq \mathcal{Q}$ such that, for all word w of length l in A^*, the CA $\mathcal{A}$ recognizes w with accepting states $\mathcal{Q}_{acc}$ in time $T(l)$ if and only if $w \in L$.*

3.3 Real-Time

Definition 34 *Given an r-complete neighborhood V, we define the* real-time function *on V*

$$\begin{aligned} TR_V : \mathbb{N} &\to \mathbb{N} \\ l &\mapsto \min\{t \in \mathbb{N} | [\![0, l-1]\!] \subseteq V^t\} \end{aligned}$$

Definition 35 *Let $\mathcal{A} = (\mathcal{Q}, V, f)$ be a CA where V is r-complete and L a language on $A \subseteq \mathcal{Q}$. We'll say that the CA $\mathcal{A}$ recognizes L in* real-time *if it recognizes L in time TR_V.*

4 Constant Speed-Up Theorem

In this section we will prove the following theorem

Theorem 41 *Let V be an r-complete neighborhood. For all $k \in \mathbb{N}$, if a language L can be recognized by a CA working on the neighborhood V in time $TR_V + k$ then it is recognized by a CA working on V in real-time.*

To prove this theorem, let's consider an r-complete neighborhood V (and all the corresponding notations from section 2). Let L be a language on a given alphabet recognized by a CA $\mathcal{A}$ working on V in time $TR_V + k$. Let $\mathcal{Q}$ be the set of states of $\mathcal{A}$.

We will construct a CA $\mathcal{A}'$ working on V that will simulate the evolution of $\mathcal{A}$ but with a constant speed-up (of k generations).

In all the following proof, we will consider the evolution of the automaton $\mathcal{A}$ from the initial configuration corresponding to a word $w = w_0 \dots w_{l-1}$ of length l. The initial configuration is

$$\dots \# \# \# w_0 w_1 \dots w_{l-1} \# \# \# \dots$$

The state of the cell $c \in \mathbb{Z}$ at time t in the evolution of $\mathcal{A}$ will be noted $\langle c \rangle_t$. The states of $\mathcal{A}'$ will be tuples of elements of $\mathcal{Q}$.

4.1 The Evolution of $\mathcal{A}'$

Time 0. The initial configuration is the same than the one of $\mathcal{A}$. Each cell c is in the state $\langle c \rangle_0$.

Time $0 \rightarrow (t_c + t_s)$. The cells will gather information, meaning that they will only read the state of their neighbors (the transition rule of $\mathcal{A}$ is not applied) and memorize the state of the cells that are further away from them.

Time $(t_c + t_s)$. Each cell c knows exactly the states $\{\langle c + x \rangle_0 | x \in V^{t_c + t_s}\}$, and will "assume" that all states $\{\langle c + x \rangle_0 | x \in [\![t_s x_p, (t_c + t_s + k) x_p]\!]\}$ that it doesn't already know (the ones not in $c + V^{t_c + t_s}$) are $\#$. Note that this assumption is true for the cells that are close enough to the end of the word w, and false for the others (we'll see this more in details later). Also each cell will do the symmetrical assumption that all states in $\{\langle c - x \rangle_0 | x \in [\![t_s x_n, (t_c + t_s + k) x_n]\!]\}$ that it doesn't know are $\#$.

From now on we'll make a difference between the information that a cell "knows" and the information it "assumes" on its right and on its left.

Time $(t_c + t_s) \rightarrow \infty$. Here, each cell will apply the transition rule of $\mathcal{A}$ to all the information it knows. This way, at time $(t_c + t_s + t)$ each cell knows the states $\{\langle c + x \rangle_t | x \in V^{t_c + t_s}\}$. Also, it will apply the transition rule to the assumed information. In this computation, the cell might have incompatible information (what it has assumed so far and what the cell $(c + x_p)$ knows or assumes for example). In this case, it will always give the priority to the information held by the cell $(c + x_p)$ when computing assumptions on its right, and to the information held by $(c - x_n)$ when computing assumptions on its left.

4.2 Why the Simulation Works

Here we will prove that the automaton $\mathcal{A}'$ described in the previous subsection recognizes the language L in real-time. We will still focus on the evolution from the word w and keep the notations of the states in the evolution of $\mathcal{A}$.

Claim 41 *At time $(t_c + t_s)$ the cell c knows the states $\{\langle c+x\rangle_0 | x \in V^{t_c+t_s}\}$.*

Proof. By induction, if at time t the cell knows $\{\langle c+x\rangle_0 | x \in V^t\}$ then it can see its neighbors $(V + c)$ and their stored information. At time $(t + 1)$ it can therefore have the information

$$\{\langle c+v+x\rangle_0 | v \in V, x \in V^t\} = \{\langle c+x\rangle_0 | x \in V^{t+1}\}$$

□

Claim 42 *At time (t_c+t_s+t) the cell c knows the states $\{\langle c+x\rangle_t | x \in V^{t_c+t_s}\}$.*

Proof. Induction again. To compute the states

$$\{\langle c+x\rangle_{t+1} | x \in V^{t_c+t_s}\}$$

the cell c needs to see the states

$$\{\langle c+x\rangle_t | x \in (V^{t_c+t_s} + V)\}$$

which it does by looking at its neighbors in $(V + c)$. □

Definition 41 *At a given time t, a cell c will be called* r-correct *if all the assumptions it does on its right are correct. The cell will be called* r-incorrect *otherwise.*

We have similar definitions of l-correct *and* l-incorrect *for the left side.*

Remark. If at time (t_c+t_s+t) the cell c is r-correct, then it knows or assumes correctly all the states $\{\langle c+x\rangle_t | x \in [\![-t_s x_n, (t_c+t_s+k)x_p]\!]\}$.

Claim 43 *At time $(t_c + t_s)$ all the cells $c \geq (l - t_s x_p - x_0)$ are r-correct.*

Proof. These cells assume that the initial states of the cells $c \geq l$ are # which is true according to the initial configuration (by definition $(t_s x_p + x_0)$ is the smallest positive integer not in $V^{t_c+t_s}$) (see fig. 3). □

Claim 44 *If the cell $(c + x_p)$ is r-correct at time $(t_c + t_s + t)$ then the cell c is r-correct at time $(t_c + t_s + t + 1)$.*

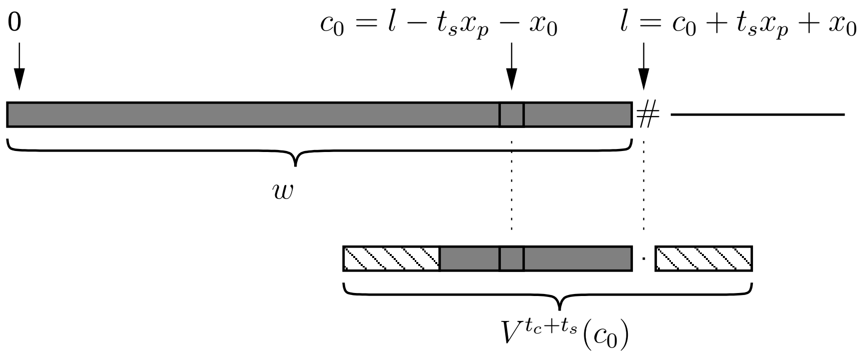

Fig. 3. Proof of claim 43

Proof. We have seen that when the cell c applies the transition rule to the assumed information, it considers the information held by $(c+x_p)$ with a higher priority. To compute correctly the states

$$\{\langle c+x\rangle_{t+1} | x \in [\![t_s x_p, (t_c+t_s+k)x_p]\!]\}$$

the cell c needs to see the states

$$\{\langle c+x+v\rangle_t | v \in V, x \in [\![t_s x_p, (t_c+t_s+k)x_p]\!]\}$$

that are all included in

$$\{\langle c+x_p+x\rangle_t | x \in [\![-t_s x_n, (t_c+t_s+k)x_p]\!]\}$$

so c sees all the correct information at time (t_c+t_s+t) and is therefore r-correct at time (t_c+t_s+t+1). □

Claim 45 *At time (t_c+t_s+t), all cells $c \geq l-(t_s+t)x_p-x_0$ are r-correct.*

Proof. Immediate consequence of claims 43 and 44. □

Claim 46 *At every time, the origin and all negative cells are l-correct.*

Proof. The proof is easy and similar to the previous ones □

Claim 47 *If the length of the word w is such that $l \geq (t_s+1)x_p+x_0+1$ then the real-time (for this word) is $TR_V(l) \geq t_c + \left\lfloor \frac{l-1-x_0}{x_p} \right\rfloor + 1$.*

Proof. Let's define $\alpha = \left\lfloor \frac{l-1-x_0}{x_p} \right\rfloor$. We have $\alpha x_p + x_0 \leq l-1$. Moreover, since $l \geq (t_s+1)x_p+x_0+1$ we have $\alpha \geq t_s$ and so by theorem 21 we have

$$(\alpha x_p + x_0) \notin V^{t_c+\alpha}$$

which means that $TR_V(l) \geq t_c+\alpha+1$. □

Claim 48 *If the length of w is such that $l \geq (t_s+1)x_p + x_0 + 1$ then at time $TR_V(l)$ the automaton $\mathcal{A}'$ can determine whether or not the word w is in L.*

Proof. From claim 45, we know that the origin is r-correct at times

$$t \geq t_c + \left\lceil \frac{l - x_0}{x_p} \right\rceil = t_c + \left\lfloor \frac{l - 1 - x_0}{x_p} \right\rfloor + 1$$

This means that the origin is r-correct at time $TR_V(l)$ (from claim 47) and, since it is always l-correct (claim 46), at time $TR_V(l)$ the origin knows (or assumes correctly) all the states in

$$\{\langle c\rangle_{TR_V(l)-t_c-t_s} | c \in [\![-(t_c+t_s+k)x_n, (t_c+t_s+k)x_p]\!]\}$$

which contain all the necessary information to compute the state $\langle 0\rangle_{TR+k}$. The automaton can therefore determine if w is in L. $\square$

This last proposition and the fact that there is only a finite number of words of length $l \leq (t_s+1)x_p + x_0$ (so the automaton $\mathcal{A}'$ can handle these exceptions directly) end the proof of theorem 41.

5 Real-Time Equivalences

Definition 51 *For every neighborhood V and every natural number n, we will define $L^n_{TR}(V)$ the set of languages on the alphabet $[\![0, n-1]\!]$ that can be recognized in real-time by a CA working on V. We will also define*

$$L_{TR}(V) = \bigcup_{n\in\mathbb{N}} L^n_{TR}(V)$$

This section is dedicated to the proof of the following theorem.

Theorem 51 *Let V and V' be two neighborhoods, we have the following relations:*

1. *if V is r-complete and V' is complete then $L_{TR}(V) \subseteq L_{TR}(V')$;*
2. *if V and V' are complete then $L_{TR}(V) = L_{TR}(V')$;*
3. *if V and V' are r-incomplete then $L_{TR}(V) = L_{TR}(V')$.*

Proposition 51 *For all r-complete neighborhood V, if $-x_n = \min V$ and $x_p = \max V$ then $L_{TR}(V) = L_{TR}([\![-x_n, x_p]\!])$.*

Proof. It is obvious that any automaton on V can be simulated by an automaton on $[\![-x_n, x_p]\!]$ without any loss in time. Proposition 22 shows that the difference between the real-time on V and the real-time on $[\![-x_n, x_p]\!]$ is at most t_c for any word. Therefore, from theorem 41, we have $L_{TR}(V) \subseteq L_{TR}([\![-x_n, x_p]\!])$.

Now if a language L is recognized in real-time by an automaton working on $[\![-x_n, x_p]\!]$ we can construct an automaton working on V that will gather information during the first t_c steps. From there each cell sees enough information at each step to apply the transition rule of the automaton working on $[\![-x_n, x_p]\!]$ and update its information. The resulting automaton recognizes L in time $TR_{[\![-x_n,x_p]\!]} + t_c$ which is obviously smaller than $TR_V + t_c$, and so by theorem 41 we have the converse inclusion. □

Proposition 52 *For all r-complete neighborhood V and all $k \in \mathbb{N}^*$, we have $L_{TR}(V) = L_{TR}(V^k)$.*

Proof. By proposition 51 we can restrict ourself to the connex neighborhoods. It is clear that for a connex neighborhood V, $TR_V \geq k(TR_{V^k} - 1)$. We know that one step of a CA working on V^k can be simulated by a CA working on V in k steps, so every language L that can be recognized in time TR_{V^k} on V^k can be recognized in time $kTR_{V^k} \leq TR_V + k$. From theorem 41 we have $L_{TR}(V^k) \subseteq L_{TR}(V)$.

Reciprocally, it is easy to see that $TR_{V^k} \geq \lfloor TR_V/k \rfloor$. Since we know that a CA working on V^k can simulate k steps of a CA working on V in only one generation, every language that can be recognized on V in real-time can be recognized on V^k in time $\lceil TR_V/k \rceil \leq TR_{V^k} + 1$. Conclusion comes from theorem 41. □

Proposition 53 *For all $x_n \leq x'_n \in \mathbb{N}$ and $x_p \in \mathbb{N}^*$, we have*

$$L_{TR}([\![-x_n, x_p]\!]) \subseteq L_{TR}([\![-x'_n, x_p]\!])$$

Proof. It is enough to observe that the real-time functions for both of these neighborhoods are the same, and since it is obvious that every CA working on the smaller can be simulated without loss of time by a CA working on the bigger we have this trivial inclusion. □

Proposition 54 *For all x_n and x_p in $\mathbb{N}^*$ we have*

$$L_{TR}([\![-x_n, x_p]\!]) = L_{TR}([\![-2x_n, x_p]\!])$$

Proof. It is well known that if there is a CA working on $[\![-2x_n, x_p]\!]$ that recognizes L in real-time, then there exists another CA working on the same neighborhood that recognizes L in real-time on limited space (meaning that a cell in state # never changes its state). This technique is very similar to the technique used to prove that Turing machines working on a semi-infinite tape are equivalent to the ones working on a bi-infinite tape.

To prove the inclusion $L_{TR}([\![-2x_n, x_p]\!]) \subseteq L_{TR}([\![-x_n, x_p]\!])$, consider an automaton $\mathcal{A}$ working on $V_2 = [\![-2x_n, x_p]\!]$ that recognizes a language L in real-time and limited space. Let $w = w_0 w_1 \dots w_{l-1}$ be a word, and let's consider the evolution of $\mathcal{A}$ from the initial configuration corresponding to w. Like previously, we will denote as $\langle c \rangle_t$ the state of the cell c at time t in the evolution of $\mathcal{A}$.

Now we will explain the behaviour of a CA working on the neighborhood $V_1 = [\![-x_n, x_p]\!]$ starting from the same initial configuration, and recognizing the language L in exactly the same time as $\mathcal{A}$ (the evolution of $\mathcal{A}'$ on an initial configuration corresponding to a word of length 7 for $V_1 = [\![-1, 1]\!]$ is illustrated on figure 4).

– Time 0. Each cell c has the information $\langle c\rangle_0$.
– Time 1. Cell c has the information $\langle c\rangle_0$ and $\langle c+1\rangle_0$. This is possible because $\{0,1\} \subseteq V$.
– When $1 \leq t \leq c$, the cell c has the information $\langle c+t-1\rangle_0$ and $\langle c+t\rangle_0$. This again is possible simply by looking at the cell $(c+1)$.
– When $t = c+k$ with $0 \leq k \leq x_n$, the cell c has information $\langle 2c-1\rangle_0$, $\langle 2c-1\rangle_1$, $\ldots$, $\langle 2c-1\rangle_k$ and $\langle 2c\rangle_0$, $\langle 2c\rangle_1$, $\ldots$, $\langle 2c\rangle_k$. The information up to time $(k-1)$ is already on c at the previous time (only memory), and the information of time k can be computed from the information c sees because by construction it can now see "twice as far" to the left, from the compression.
– At time $(c + x_n + t)$, the cell c has the information $\langle 2c-1\rangle_t$, $\langle 2c-1\rangle_{t+1}$, $\ldots$, $\langle 2c-1\rangle_{t+x_n}$ and $\langle 2c\rangle_t$, $\langle 2c\rangle_{t+1}$, $\ldots$, $\langle 2c\rangle_{t+x_n}$. Again, some information was already on c at the previous time, the rest can be computed.

With such an evolution, we can see that the origin 0 has at each step t the information $\langle 0\rangle_t$ so the automaton $\mathcal{A}'$ can decide whether or not w is in L at time $TR_{V_1}(l) = TR_{V_2}(l)$, which proves the inclusion. The converse inclusion is covered by proposition 53. □

Fig. 4. The evolution of $\mathcal{A}'$ (time goes from bottom to top)

…	#	$\langle 0\rangle_6\langle 0\rangle_5$	$\langle 1\rangle_5\langle 2\rangle_5$ $\langle 1\rangle_4\langle 2\rangle_4$	$\langle 3\rangle_4\langle 4\rangle_4$ $\langle 3\rangle_3\langle 4\rangle_3$	$\langle 5\rangle_3\langle 6\rangle_3$ $\langle 5\rangle_2\langle 6\rangle_2$	#	…			
…	#	$\langle 0\rangle_5\langle 0\rangle_4$	$\langle 1\rangle_4\langle 2\rangle_4$ $\langle 1\rangle_3\langle 2\rangle_3$	$\langle 3\rangle_3\langle 4\rangle_3$ $\langle 3\rangle_2\langle 4\rangle_2$	$\langle 5\rangle_2\langle 6\rangle_2$ $\langle 5\rangle_1\langle 6\rangle_1$	#	…			
…	#	$\langle 0\rangle_4\langle 0\rangle_3$	$\langle 1\rangle_3\langle 2\rangle_3$ $\langle 1\rangle_2\langle 2\rangle_2$	$\langle 3\rangle_2\langle 4\rangle_2$ $\langle 3\rangle_1\langle 4\rangle_1$	$\langle 5\rangle_1\langle 6\rangle_1$ $\langle 5\rangle_0\langle 6\rangle_0$	#	…			
…	#	$\langle 0\rangle_3\langle 0\rangle_2$	$\langle 1\rangle_2\langle 2\rangle_2$ $\langle 1\rangle_1\langle 2\rangle_1$	$\langle 3\rangle_1\langle 4\rangle_1$ $\langle 3\rangle_0\langle 4\rangle_0$	$\langle 5\rangle_0\langle 6\rangle_0$	$\langle 6\rangle_0$#	#	…		
…	#	$\langle 0\rangle_2\langle 0\rangle_1$	$\langle 1\rangle_1\langle 2\rangle_1$ $\langle 1\rangle_0\langle 2\rangle_0$	$\langle 3\rangle_0\langle 4\rangle_0$	$\langle 4\rangle_0\langle 5\rangle_0$	$\langle 5\rangle_0\langle 6\rangle_0$	$\langle 6\rangle_0$#	#	…	
…	#	$\langle 0\rangle_1\langle 0\rangle_0$	$\langle 1\rangle_0\langle 2\rangle_0$	$\langle 2\rangle_0\langle 3\rangle_0$	$\langle 3\rangle_0\langle 4\rangle_0$	$\langle 4\rangle_0\langle 5\rangle_0$	$\langle 5\rangle_0\langle 6\rangle_0$	$\langle 6\rangle_0$#	#	…
…	#	$\langle 0\rangle_0$	$\langle 1\rangle_0$	$\langle 2\rangle_0$	$\langle 3\rangle_0$	$\langle 4\rangle_0$	$\langle 5\rangle_0$	$\langle 6\rangle_0$	#	…

To prove the 3 propositions of theorem 51, let us consider two r-complete neighborhoods V and V', with $-x_n = \min V$, $-x'_n = \min V'$, $x_p = \max V$ and $x'_p = \max V'$.

1. If V' is complete then $x'_n > 0$. From propositions 51, 52, 53 and 54 we have, for some k big enough

$$\begin{aligned} L_{TR}(V) = L_{TR}([\![-x_n, x_p]\!]) = L_{TR}([\![-x'_p x_n, x'_p x_p]\!]) &\subseteq L_{TR}([\![-2^k x_p x'_n, x_p x'_p]\!]) \\ \subseteq L_{TR}([\![-x_p x'_n, x_p x'_p]\!]) \subseteq L_{TR}([\![-x'_n, x'_p]\!]) &\subseteq L_{TR}(V') \end{aligned}$$

2. This equality is a direct consequence of the previous inclusion.
3. If both are r-incomplete then $x_n = x'_n = 0$ and

$$L_{TR}(V) = L_{TR}([\![0, x_p]\!]) = L_{TR}([\![0, x'_p x_p]\!]) = L_{TR}([\![0, x'_p]\!]) = L_{TR}(V')$$

This ends the proof of theorem 51.

6 Conclusion

We have here proven an extension of Cole's equivalence [3] for one-dimensional neighborhoods. Since the usual simulation between neighborhoods is linear (in time), it was already known that linear, polynomial and exponential capabilities of the complete neighborhoods are all equal. We now know that the real-time capabilities are the same.

We have also studied an extension of one-way cellular automata: the CA working on r-incomplete neighborhoods. For these neighborhoods too we have obtained a strong equivalence. These two kinds of neighborhoods (complete and r-incomplete) are the only ones that enable a computation power equivalent to Turing machines. On the other neighborhoods (not r-complete) it is possible to prove that some cells will never interact with the origin (no matter what their state is) and that the position of these “invisible cells” is ultimately periodic. From this observation, it is easy to show that the languages that can be recognized on these neighborhoods are exactly the sub-class of Turing languages that do not depend on the letters on the “invisible cells”. We can prove a constant speed-up theorem for this sub-class too.

We have therefore shown that language recognition by one-dimensional cellular automata can always be reduced to a recognition on either the usual neighborhood $\{-1, 0, 1\}$ or the one-way neighborhood $\{0, 1\}$.

In dimension 2 and above, it is known that the different neighborhoods aren't equivalent in terms of real-time recognition (of two-dimensional shapes) [14]. Although we can prove a theorem similar to theorem 21 that gives an exact description of any iteration of a two-dimensional neighborhood, it is unknown if we have a constant speed-up theorem.

References

1. Albert, J., Čulik II, K.: A simple universal cellular automaton and its one-way and totalistic version. Complex Systems **1** (1987) 1–16
2. Choffrut, C., Čulik II, K.: On real-time cellular automata and trellis automata. Acta Informatica **21** (1984) 393–407
3. Cole, S.N.: Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Transactions on Computers **C-18** (1969) 349–365
4. Čulik, K., Hurd, L.P., Yu, S.: Computation theoretic aspects of cellular automata. Phys. D **45** (1990) 357–378
5. Delorme, M., Mazoyer, J.: Reconnaissance parallèle des langages rationnels sur automates cellulaires plans. Theor. Comput. Sci. **281** (2002) 251–289
6. Fischer, P.C.: Generation of primes by one-dimensional real-time iterative array. Journal of the Assoc. Comput. Mach. **12** (1965) 388–394
7. Ibarra, O., Jiang, I.: Relating the power of cellular arrays to their closure properties. Theoretical Computer Science **57** (1988) 225–238
8. Martin, B.: A universal automaton in quasi-linear time with its s-n-m form. Theoretical Computer Science **124** (1994) 199–237
9. Mazoyer, J., Reimen, N.: A linear speed-up theorem for cellular automata. Theor. Comput. Sci. **101** (1992) 59–98
10. Smith III, A.R.: Simple computation-universal cellular spaces. J. ACM **18** (1971) 339–353
11. Smith III, A.R.: Real-time language recognition by one-dimensional cellular automata. Journal of the Assoc. Comput. Mach. **6** (1972) 233–253
12. Terrier, V.: Language recognizable in real time by cellular automata. Complex Systems **8** (1994) 325–336
13. Terrier, V.: Language not recognizable in real time by one-way cellular automata. Theoretical Computer Science **156** (1996) 281–287
14. Terrier, V.: Two-dimensional cellular automata and their neighborhoods. Theor. Comput. Sci. **312** (2004) 203–222
15. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana, IL, USA (1966)