# L-Convex Polyominoes Are Recognizable in Real Time by 2D Cellular Automata

Anaël Grandjean, Victor Poupet

# L-Convex Polyominoes are Recognizable in Real Time by 2D Cellular Automata

Anaël Grandjean and Victor Poupet

LIRMM, Université Montpellier 2
161 rue Ada, 34392 Montpellier, France
victor.poupet@lirmm.fr, anael.grandjean@lirmm.fr

**Abstract.** A polyomino is said to be L-convex if any two of its cells are connected by a 4-connected inner path that changes direction at most once. The 2-dimensional language representing such polyominoes has been recently proved to be recognizable by tiling systems by S. Brocchi, A. Frosini, R. Pinzani and S. Rinaldi. In an attempt to compare recognition power of tiling systems and cellular automata, we have proved that this language can be recognized by 2-dimensional cellular automata working on the von Neumann neighborhood in real time.
Although the construction uses a characterization of L-convex polyominoes that is similar to the one used for tiling systems, the real time constraint which has no equivalent in terms of tilings requires the use of techniques that are specific to cellular automata.

## Introduction

Two-dimensional cellular automata and tiling systems are two different models that can be considered to recognize classes of two-dimensional languages (or picture languages). Although they share some similarities such as locality and uniformity, the two models are fundamentally different.

Tiling systems as language recognizers were introduced by D. Giammarresi and A. Restivo in 1992 [3] and are based on the model of tile sets introduced by H. Wang [7]. The strength of the model lies in its inherent non-determinism. The system itself is a set of local rules describing valid image patterns and a picture language is recognized by the system if it is the image by a projection of the set of configurations that verify all local rules.

Cellular automata on the contrary 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 automata studied by Ulam and von Neumann were 2-dimensional) and are therefore particularly well suited to 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.

Because tiling systems lack dynamic behavior, some picture languages that can be recognized by cellular automata with minimal space and time complexity (in real time) cannot be recognized by tiling systems, such as the language of square pictures with vertical symmetry.

Conversely, the non-determinism of tiling systems should allow the recognition of languages that cannot be recognized by cellular automata in low time complexities. It is straightforward for instance to verify that the language considered in [5] as an example of language that cannot be recognized in real time by a cellular automaton working on the Moore neighborhood but can be recognized on the von Neumann neighborhood can be recognized by a tiling system, thus proving that tiling systems and real time cellular automata on the Moore neighborhood are incomparable.

Because the language of L-convex polyominoes was recently proved to be recognizable by tiling systems when it was previously though not to be, we decided to investigate its recognizability by real time von Neumann neighborhood cellular automata. Although the language was also recognized by cellular automata, the construction turned out to be quite different from the case of tiling systems and used some techniques specific to cellular automata (and possibly von Neumann neighborhood cellular automata). This article describes said construction.

## 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* neighborhood *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 neighborhood 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 $\delta$ 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 dynamic, 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 $\mathcal{A}$, and 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 Neighborhoods).** *In $d$ dimensions, the most commonly considered neighborhoods are the von Neumann neighborhood $\mathcal{N}_{\mathrm{vN}} = \{c \in \mathbb{Z}^d,\ ||c||_1 \leq 1\}$ and the Moore neighborhood $\mathcal{N}_{\mathrm{M}} = \{c \in \mathbb{Z}^d, ||c||_\infty \leq 1\}$. Figure 1 illustrates these two neighborhoods in 2 dimensions.*

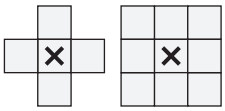

**Fig. 1.** The von Neumann (left) and Moore (right) neighborhoods in 2 dimensions.

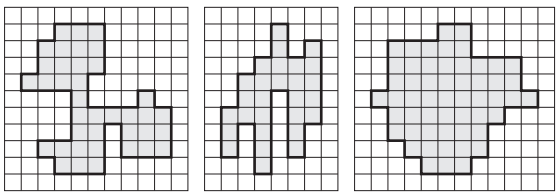

**Fig. 2.** Three polyominoes. The center and right ones are vertically convex, the right one is HV-convex.

### 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 4 (Picture).** *For $n, m \in \mathbb{N}$ and $\Sigma$ a finite alphabet, an $(n,m)$-*picture *(picture of width $n$ and height $m$) over $\Sigma$ 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 $\Sigma$ and $\Sigma^{*,*} = \bigcup_{n,m\in\mathbb{N}} \Sigma^{n,m}$ the set of all pictures over $\Sigma$. A* picture language *over $\Sigma$ is a set of pictures over $\Sigma$.*

**Definition 5 (Picture Configuration).** *Given an $(n,m)$-picture $p$ over $\Sigma$, 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 6 (Picture Recognizer).** *Given a picture language $L$ over an alphabet $\Sigma$, 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,*

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

Because cellular automata work with a finite neighborhood, 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 7 (Real Time).** *Given a neighborhood $\mathcal{N} \subset \mathbb{Z}^d$ 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 neighborhood, 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 neighborhood can also be recognized in real time by a 2DCA working on the von Neumann neighborhood.*

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.

### 1.3 Polyominoes

**Definition 8 (Polyomino).** *A* placed polyomino *is a finite and 4-connected subset of $\mathbb{Z}^2$. A* polyomino *is the equivalence class of a placed polyomino up to translation.*

**Definition 9 (HV-Convexity).** *A polyomino $p$ is said to be* horizontally (resp. vertically) convex *if any cell between two cells of the polyomino on a same horizontal (resp. vertical) line is also a cell of the polyomino :*

$$\forall x_1, x_2, x_3, y \in \mathbb{Z}, \quad x_1 \leq x_2 \leq x_3 \wedge (x_1, y) \in p \wedge (x_3, y) \in p \Rightarrow (x_2, y) \in p$$

*A polyomino is* HV-convex *if it is both horizontally and vertically convex (see Figure 2).*

We will now present the notion of L-convex polyomino, first introduced in [2] to classify HV-convex polyominoes. Informally, an L-convex polyomino $p$ is such that for any two of its cells there exists a 4-connected path of cells of $p$ that connects them such that the path changes direction at most once (see Figure 3).

The following remarks will lead to a formal definition of L-convex polyominoes :

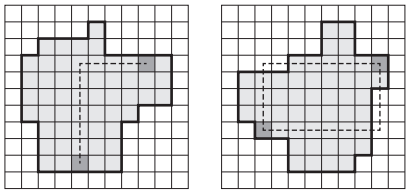

**Fig. 3.** The polyomino on the left is L-convex (the figure shows an inner path connecting two cells with at most one direction change, and there is such a path for any pair of cells). The polyomino on the right is HV-convex but not L-convex as illustrated by the pair of highlighted cells for which there is no inner connecting path that changes direction at most once.

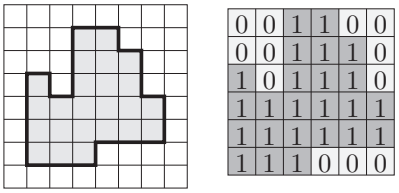

**Fig. 4.** A polyomino (left) and its corresponding picture over $\{0, 1\}$ (right). When this picture is encoded as a configuration of a cellular automaton, the origin of the automaton is on the lower left corner of the picture.

- a path that changes direction at most once connecting two cells of a polyomino $p$ on the same row (resp. column) is fully horizontal (resp. vertical) therefore L-convex polyominoes are HV-convex ;
- if $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ are two cells in an L-convex polyomino $p$, either $a_1 = (x_1, y_2)$ or $a_2 = (x_2, y_1)$ is a cell of $p$ because $a_1$ and $a_2$ are the angles of the only two paths connecting $c_1$ and $c_2$ that change direction at most once ;
- if a polyomino $p$ is HV-convex and such that for any two of its cells $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ either $a_1 = (x_1, y_2)$ or $a_2 = (x_2, y_1)$ is a cell of $p$, then $p$ is L-convex since by HV-convexity, the whole path connecting $c_1$ to $c_2$ going through $a_1$ or $a_2$ is in $p$.

**Definition 10 (L-Convexity).** *A polyomino $p$ is L-convex if it is HV-convex and verifies*

$$\forall x_1, x_2, y_1, y_2 \in \mathbb{Z}, \quad (x_1, y_1) \in p \wedge (x_2, y_2) \in p \Rightarrow (x_1, y_2) \in p \vee (x_2, y_1) \in p$$

Given a polyomino $p$, the picture over the alphabet $\{0, 1\}$ associated to $p$ is the picture whose dimensions are the dimensions of the minimal bounding rectangle of $p$, where the cell has state 1 if the corresponding cell is in the polyomino and 0 otherwise (see Figure 4). We define the language $L_{\text{L-convex}}$ as the picture language of all L-convex polyomino pictures.

## 2 Main Result

This section will be entirely devoted to the proof of the following result

**Theorem 1.** *The picture language $L_{\text{L-convex}}$ of L-convex polyomino pictures is recognizable in real time by a 2DCA working on the von Neumann neighborhood.*

The proof will be done by describing the behavior of a 2DCA working on the von Neumann neighborhood that recognizes $L_{\text{L-convex}}$ in real time. In this description we will use cardinal directions north, south, east and west to denote the different directions on the configuration as follows :

- north is towards the increasing $y$ axis ;
- south is towards the decreasing $y$ axis ;
- east is towards the increasing $x$ axis ;
- west is towards the decreasing $x$ axis.

With such conventions, the origin of the automaton is located at the south-west (SW) angle of the picture in the initial configuration and the picture therefore extends from the origin eastward and northward.

### 2.1 Preliminary Check

First of all, the automaton must check that the input is the picture of a HV-convex polyomino.

To do so, during the first step of the computation, each cell containing a 1 considers its neighbors and remembers which of them also contains a 1. Then a signal moves westward from the eastmost point of each row and southward from the northmost point on each column. These signals check that each row and each column contains exactly one segment of connected 1 symbols. Moreover, the signals check that the segment of 1 on each line and column is connected to that of the neighbor rows and columns using the neighboring information gathered during the first step.

These two properties guarantee that the polyomino is connected, HV-convex and that the picture's dimensions are that of the minimal bounding rectangle (no empty row or column). If an error is found on a row or column, the signal is directed towards the origin and the input is not accepted.

We can now assume that the input corresponds to a HV-convex polyomino picture, and must determine whether it is also L-convex.

### 2.2 Characterization of L-Convex Polyominoes

We will now present the characterization of L-convex polyominoes that will be used by the automaton. It is a slighly rephrased version of the characterization presented in [1] (Theorem 2).

Given a polyomino $p$, we say that a cell of $p$ is a *corner* if it has two consecutive neighbors that are not in $p$. We classify corners depending on the directions in which such neighbors not in $p$ are located : a north-east (NE) corner is one such that the northern and eastern neighbors are not in $p$, and we similarly have NW , SW and SE corners (see Figure 5 for an illustration of NE corners). Note that corner types are not exclusive : a cell can for instance be both a NE and NW corner.

**Proposition 2 (Characterization of L-convex polyominoes [1]).** *A HV-convex polyomino $p$ is L-convex if and only if for every NE corner $c = (x, y)$, denote by $(x, y')$ the southest cell of $p$ on the same column as $c$, and $(x', y)$ the westmost cell of $p$ on the same row as $c$, there is no cell $(x'', y'')$ of $p$ verifying any of the following three conditions*

*(a).* $x'' > x$ *(resp.* $x'' > x'$*) and* $y'' < y'$
*(b).* $x'' < x'$ *(resp.* $x'' < x'$*) and* $y'' > y'$
*(c).* $x'' < x'$ *(resp.* $x'' < x'$*) and* $y'' < y'$

*and the symmetric conditions holds for all NW corners (in the South and East directions).*

Figure 5 illustrates this characterization.

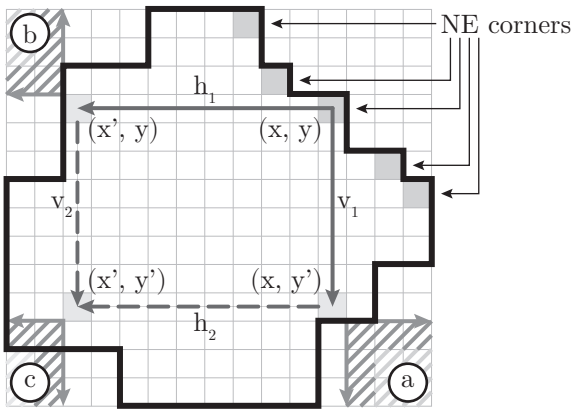


**Fig. 5.** A HV-convex polyomino is L-convex is for any of its NE corners (represented as dark grey cells), no cell of the polyomino lies in any of the three zones represented in hatched light grey, and symmetrically for all of its NW corners. The illustrated polyomino is not L-convex because there are two cells in the lower left hatched area (these cells cannot be connected to the represented NE corner by an inner path with at most one direction change).

*Proof (sketch).* It is enough to verify that all pairs of corners of a HV-convex polyomino are connected by an inner path with at most one change of direction. Moreover by symmetry we can consider only NW and NE corners.

The cells $(x', y)$, $(x, y')$ and $(x', y')$ in the characterization represent the farthest points that can be reached from a given corner in their respective directions. Cells of the three restricted areas cannot be connected to the corner and conversely all cells not in these areas can be connected to the corner.

Note that because the polyomino is assumed to be HV-convex it is enough to check that there is no polyomino cell on the two lines extending from the starting

check point (represented in dark hatched grey in Figure 5). For instance, for the condition $(a)$, it is enough to check that there is no cell $(x'', y' - 1)$ with $x'' > x$ and no cell $(x + 1, y'')$ with $y'' < y'$ in the polyomino. This follows from the 4-connectedness of the polyomino.

Although the conditions to verify are perfectly symmetric for NE and NW corners, when implementing it on a real time cellular automaton the case of NE corners is significantly simpler because all signals move towards the origin at maximum speed so the result of the verification easily arrives on time. On the other hand, for NW corners, some signals move eastward (away from the origin) so it would take too much time to send the signal all the way to the east side and back to the origin. We will therefore now focus on implementing the characterization for NE corners and come back to the NW corners at the end of the proof.

### 2.3 Compression and Marking

The characterization from Proposition 2 depends on cells being able to tell if there is a polyomino cell in a given direction from them. To make sure that each cell knows this information, consider signals going eastward from the west side of each row. If the initial configuration is the picture configuration of a HV-convex polyomino, there is exactly one segment of 1 symbols on each row. Before the signal meets the first 1, cells can be notified that there is no 1 westward and that there is at least one 1 eastward. On the segment of 1, cells are notified of whether they are a border cell or an inner cell and, after the segment of 1, all cells are notified that there is a 1 westward and none eastward. Of course the same thing can be done on columns with northward signals.

Now consider a horizontal compression of the input as illustrated by Figure 6. To compress the input, consider that each cell can now hold two initial states instead of one (this can be done by increasing the number of states of the automaton) and move all states westward unless the column in which they should go is full (contains two states) or is out of the boudaries of the initial configuration (ignore the darker dots from Figure 6 for the moment).

Such a compression takes $\lceil \frac{n}{2} \rceil$ time steps where $n$ is the width of the input. During these steps, no signal can propagate westward because the initial data is already moving west at maximum speed but the time lost performing the compression can be recovered afterwards because each cell now sees twice as many states horizontally, which means that relatively to the original states, horizontal signals can perform two steps at a time.

During the compression, signals can however be propagated eastward (as illustrated by the darker dots in Figure 6). This means that while the compression is taking place, the signal indicating to each cell if it has 1 symbols east or west can propagate, so that at the end of the compression, cells have access to this information.

By performing a vertical compression after the horizontal one we can otain in half of the real time a compressed copy of the initial configuration on which every cell now has the added information of whether there is a 1 in any of the

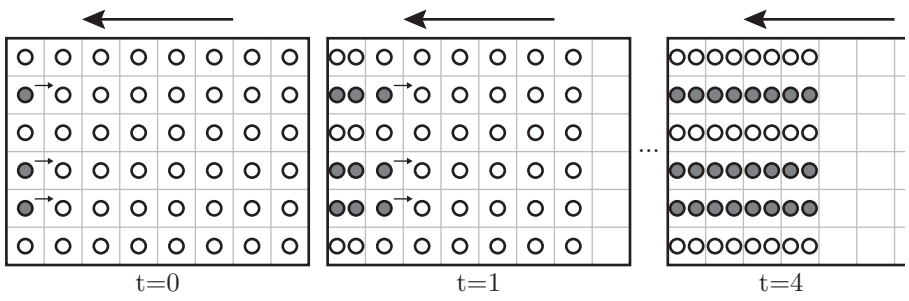


**Fig. 6.** Horizontal compression of the input, with eastward transmission of information (dark dots).

four directions. Moreover, later in the construction we will need to know which columns correspond to the same horizontal segment on the southern border of the polyomino, so we also propagate northward signals during the vertical compression from the borders of all horizontal segments of the southern border of the polyomino (see dashed northward arrows in Figure 8).

After both compressions, the computation of the automaton can properly start and in this computation horizontal and vertical signals can propagate twice as fast and all information is twice closer to the origin. This means that the compressed run of the automaton can behave exactly as if the configuration was not compressed but was given the extra information propagated by the eastward and northward signals from the beginning[1]. We will now ignore the compression in the following explanations, and simply consider that the information propagated by the northward and eastward signals is readily available to each cell.

**Remark:** As it is described, it looks as if cells should know when the compression is finished to start performing the next task (be it the second compression or the accelerated simulation of the uncompressed automaton). However, one can show that cells can *asynchronously* start the next task as soon as they have the necessary information to do so. It is sufficient to detect when all cells in their neighborhood have finished the compression to perform one step of the next task. From there, we can show that if each cell advances the following task as soon as it has enough information to do so, cells that have completed the compression early will be slowed down progressively to wait for the further cells to catch up. However, the last cells to finish the compression will never be slowed down as all other cells have the necessary information available to them. This means that by continuing the computation after the compression as soon as the information is available, all cells are at least as advanced as if all had started their computation at the time when the compression is finished, thus negating the need to synchronize all cells after the compression.

[1] This compression technique works in our case because the automaton (as it will be described later) only uses horizontal and vertical signals that change directions a bounded number of times. It is only possible to simulate two steps of the uncompressed automaton if they only involve horizontal or vertical movement, not both.

### 2.4 First Conditions of the Characterization

With the informations we have, checking conditions $(a)$ and $(b)$ of Proposition 2 is very easy as it is only a matter of sending a westward and a southward signal from each NE corner. When these signals reach the border of the polyomino, they check that there are no 1 in the corresponding area by using the information that was transmitted to each cell during the compressions. If a 1 is found where it should not be, a signal is directed towards the origin to indicate that the input should not be accepted.

There are no conflicting signals during this step because there can be at most one NE corner per column and one at most per row.

### 2.5 The Third Condition

The third condition from Proposition 2 is much more complex to implement. It requires sending a westward signal $h_1$ and a southward signal $v_1$ from each NE corner and having these signals generate secondary signals $h_2$ (westward, from the collision of $v_1$ with the border of the polyomino) and and $v_2$ (southward from the collision of $h_1$ and the border). The intersection of $h_2$ and $v_2$ indicate the cell on which condition $(c)$ should be checked, as illustrated by Figure 5.

Two problems arise when implementing this behavior :

- although $v_1$ and $h_1$ signals originating from different NE corners will never overlap, if two signals arrive on the same row or column they will produce $v_2$ or $h_2$ signals that might overlap ;
- $h_2$ signals might intersect with many $v_2$ signals, but only one of them originates from the same NE corner. It is therefore necessary to ensure that the verification of condition $(c)$ is not performed on cells that do not correspond to a valid intersection of $h_2$ and $v_2$ signals.

**Priority Rule** To solve the first problem, we use a simple priority rule : if two NE corners $c_1$ and $c_2$ are north of the same horizontal segment on the southern border of the polyomino, we can ignore the easternmost one. There are two cases to consider (illustrated by Figure 7). Assume $c_1$ lies north-west of $c_2$ :

- if the westward $h_1$ signal from $c_1$ reaches the border east of that from $c_2$ (left of Figure 7), then the area that would be checked by considering the intersection of the signals $v_2$ and $h_2$ from $c_1$ (dark grey area in the Figure) is east of the one that would be considered by the intersection from $c_2$ (light grey area) and therefore contains it entirely, which means that it is not necessary to check the area indicated from $c_2$ ;
- if on the contrary the $h_1$ signal from $c_1$ arrives west of that from $c_2$ (right of Figure 7), the HV-convexity of the polyomino guarantees that there can be no 1 in either of the two areas considered by the intersections from $c_1$ and $c_2$ since there is at least one 1 north west of where the $h_1$ signal from $c_2$ arrives, no 1 west of that point so there cannot be any 1 west and south of it. In this case, it doesn't matter which intersection is considered since neither will find a contradiction with the $(c)$ condition from Proposition 2.

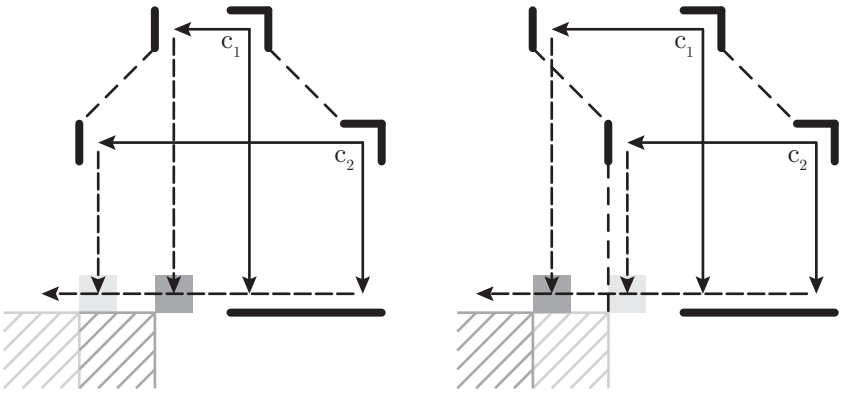


**Fig. 7.** When two $v_1$ signals arrive on the same row we can always safely ignore the one originating from the eastmost NE corner.

A symmetrical argument shows that it is sufficient to consider signals originating from the southernmost of two NE corners whose $h_1$ signals arrive on the same vertical segment of the western border of the polyomino. Horizontal and vertical signals are however handled differently because the last part of the construction is not symmetrical.

We want to make sure that there are as many $v_1$ signals as there are distinct (non-overlapping) $h_2$ signals. To do so, $v_1$ signals are not sent directly by NE corners but rather sent by the $h_1$ signal when the $h_1$ signal knows that the corner it originated from is the westmost of the corresponding horizontal segment in the southern border (see Figure 8). When an $h_1$ signal finds a cell of the polyomino north before reaching the border of the southern segment (dashed line in the figure), it knows there is another NE corner west for that segment and therefore disappears. On the contrary, if such a signal reaches the border of the southern segment it sends the $v_1$ signal southward.

**Counters** For $v_2$ signals, we need to solve the second problem that was described previously which is to determine which of the possibly many $h_2$ signals intersected is the one that originated from the same NE corner. To do so, $h_1$ signals produced by NE corners will count how many $v_2$ signals they cross while going west. If a $h_1$ signal crossed $n$ $v_2$ signals, then the $v_2$ signal it produces will consider that its corresponding $h_2$ signal is the $(n+1)$-th to last one (the last $n$ are not the one that should be considered).

Figure 9 illustrates why the result of such a behavior is correct. Consider an NE corner $c_2$ such that its $h_1$ signal crossed the $v_1$ signal produced by an NE corner $c_1$ (top circled intersection)

– if the $v_1$ signal produced by $c_2$ arrives north of the one produced by $c_1$ (left part of the figure) then the real intersection of the $v_2$ and $h_2$ signals from $c_2$

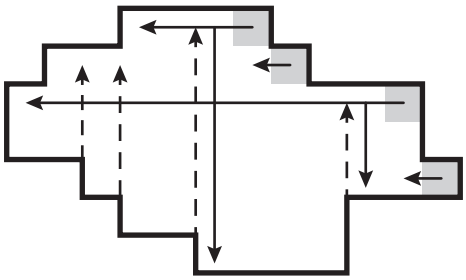

**Fig. 8.** $h_1$ signals from an NE corner are interrupted if they detect that there is another NE corner west whose $v_1$ signal would arrive on the same horizontal segment on the southern border of the polyomino. $v_1$ signals are sent by $h_1$ signals on the westmost column corresponding to the southern horizontal segment.

is the first that the $v_2$ signal from $c_2$ encounters, and the later one should be ignored (lower circled intersection) ;

– if on the contrary the $v_1$ signal from $c_2$ arrives south of that of $c_1$ (right part of the figure), the real intersection that should be considered is the last one but by considering the first the automaton will not find any contradiction to condition ($c$) since by HV-convexity of the polyomino there are no 1 south and west of either of the two intersections (so it will pick the wrong intersection but that will not change the final result).

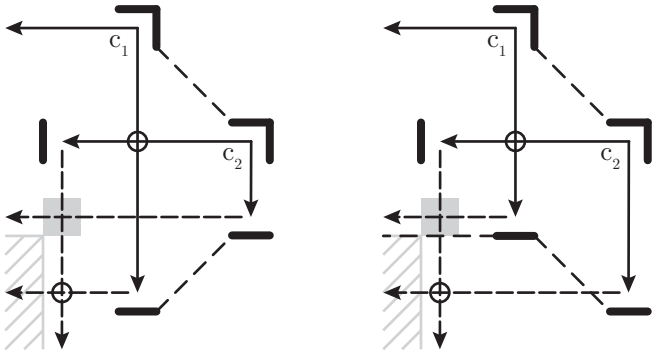


**Fig. 9.** Considering that for each $v_2$ signal crossed by the $h_1$ signal from $c_2$, one of the last intersections with an $h_2$ signal should be ignored by the subsequent $v_2$ signal leads to a correct characterization.

In order to implement this rule, signals need to carry a binary counter. This counter should follow the signal at maximal speed, and will be incremented by the $h_1$ signal for each $v_1$ signal encountered (which can be easily done as

incremental binary counters can be implemented on one-way one dimensional CA). For technical reasons, the counter has an initial value of 1.

As for the $v_2$ signal, as it crosses $h_2$ signals it checks if the area south-west of said intersection contains a 1 (which can be done instantly because of the information gathered during the initial compressions), and if so decrements the counter[2]. If the counter is equal to 0 (decreasing a 0 counter leaves it at 0) when the $v_2$ signal reaches the southernmost border of the picture, no error is detected, but if it is positive then a message is sent to the origin to indicate that the input is not L-convex.

This works because we know that if there is no 1 south-west of a cell $c$, there is no 1 south-west of a cell south of $c$. If the $h_1$ signal crosses $n$ $v_1$ signals the counter indicates $(n+1)$ when the $v_2$ signal starts moving south and if the counter is at 0 it means that the $(n+1)$ last intersections were correct according to condition $(c)$ and therefore the $(n+1)$-th to last was correct.

Checking that the counter is 0 takes $\log(n)$ steps where $n$ is the maximal value of the counter ($\log(n)$ is the maximal length of the counter). If the counter is incremented to $n$ it means that there are at least $n$ NE corners north of the one from which the signal originated. This means that if the signal moves from this corner towards the origin (south or west) at maximal speed, it would reach the origin at least $n$ steps before the real time, and therefore it can spend $\log(n)$ steps checking the value of the counter and still arrive in real time.

Moreover, conflicts of overlapping counters can be resolved by the priority rule described previously. Precedence must always be given to the counter corresponding to the southernmost NE corner :

- when an $h_1$ signal reaches the west border of the polyomino, it marks the cell on which the $v_2$ signal is produced ;
- if a $v_2$ signal moves through such a marked cell, it is erased ;
- if the counter following a $v_2$ signal is on a cell where a new $v_2$ signal is created, the counter is invalidated (the end symbol is erased) so that the new $v_2$ signal has precedence over it. An invalidated counter will ignore decrementations and will ignore the test to 0 at the end.

### 2.6 The North-West Corners

The previous subsections describe how NE corners can properly implement the characterization of L-convex polyominoes from Proposition 2. NW corners will behave in a very similar fashion, but special care must be taken to prove that the result of their verification can reach the origin in real time.

On a regular configuration, a signal issued from a NW corner needs to go east through most of the polyomino, then south and then the result of the verification should travel back west to the origin. In doing so the signal goes twice through the width of the input which cannot be done in real time. To solve

[2] Decrementation can also be performed on a binary counter moving at maximal speed but in that case the length of the counter is not reduced when going from an $2^n$ to $(2^n-1)$, but instead a leading 0 is added, which is not a problem for our construction.

this problem, we consider the path of the signal during a horizontal compression of the configuration :

- the signal starts from the NW corner on the cell $c = (x, y)$ ;
- during the $\frac{x}{2}$ first steps the signal moves west with the compression, and when the cell is compressed, the $h_1$ signal is sent eastward ;
- meanwhile, the cell $(x', y)$ that should have been the target of the $h_1$ signal moves left with the compression. The $h_1$ signal and the cell arrive at the cell $(\frac{x'}{2}, y)$ at time $(\frac{x'}{2}, y)$ ;
- the signal $v_2$ from $c$ moves south until it reaches the southern border of the input after $y$ steps. At this point a delay of at most $\log(h - y)$ steps is incurred to check the value of the counter ($h$ is the total height of the input) ;
- the result of the verification is directed towards the origin, it arrives at time $\frac{x'}{2} + y + \log(h - y) + \frac{x'}{2} < x' + h$ which is before real time.

NW corners can therefore properly perform the necessary verifications to implement the characterization from Proposition 2, which concludes the proof of Theorem 1.

## References


1. Stefano Brocchi, Andrea Frosini, Renzo Pinzani, and Simone Rinaldi. A tiling system for the class of *L*-convex polyominoes. *Theor. Comput. Sci.*, 475:73–81, 2013.
2. Giusi Castiglione and Antonio Restivo. Reconstruction of l-convex polyominoes. *Electronic Notes in Discrete Mathematics*, 12(0):290 – 301, 2003. 9th International Workshop on Combinatorial Image Analysis.
3. Dora Giammarresi and Antonio Restivo. Recognizable picture languages. *International Journal of Pattern Recognition and Artificial Intelligence*, 6(02n03):241–256, 1992.
4. Alvy R. Smith III. Real-time language recognition by one-dimensional cellular automata. *Journal of the ACM*, 6:233–253, 1972.
5. Véronique Terrier. Two-dimensional cellular automata recognizer. *Theor. Comput. Sci.*, 218(2):325–346, 1999.
6. John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, USA, 1966.
7. Hao Wang. Proving theorems by pattern recognition II. *Bell System Technical Journal*, 40:1–42, 1961.