Complex Systems **8** (1994) 325–336

Language Recognizable in Real Time by Cellular Automata

Véronique Terrier
LAIAC, Université de Caen,
Esplanade de la Paix, 14032 Caen, France

Abstract. We consider the language L of the strings over the alphabet $\{0,1\}$ of shape $x^{|x|}$ where $|x|$ is the length of the string x. We show that L is recognized in real time by an iterative array but not by a one-way cellular automaton, while the language $\{0,1\}^* \cdot L$ is recognized in real time by a cellular automaton but not by an iterative array.

1. Introduction

One-dimensional cellular automata (CAs) are known to be capable of universal computation and, as a parallel model, fast computation. A lot of attention has been devoted to CAs as language recognizers, especially to the real-time CA languages and their closure properties (see [1–8]).

Actually, it is not known if the class of real-time CA languages is closed by reversal or concatenation. Nevertheless, Ibarra and Jiang [6] have shown that the class of real-time CA languages is closed by reversal if and only if linear time and real time are equivalent; in addition, if it is closed under reversal, then it is also closed under concatenation. In fact they conjectured that the class of real-time CA languages is not closed under concatenation. In particular, they studied the real-time CA language $L = \{x^{|x|} : x \in \{0,1\}^+, |x| \geq 2\}$ (where $|x|$ denotes the length of the word x) and conjectured that $\{0,1\}^* \cdot L$ is not a real-time CA language. However, we present here a CA that recognizes this language in real time, but the problem of closure under concatenation still remains open. According to Cole [3], this language is not recognizable in real time by an iterative array (IA).

The paper is organized as follows. Section 2 recalls some useful definitions for our purpose. Section 3 shows that $\{0,1\}^* \cdot L = \{0,1\}^* \cdot \{x^{|x|} : x \in \{0,1\}^+, |x| \geq 2\}$ is a real-time CA language but not a real-time IA language. Section 4 establishes that L is a real-time IA language but not a real-time one-dimensional CA language.

2. Definitions

Cellular automata (CA). A CA is a one-dimensional array of identical finite automata (cells) numbered $1, 2, \ldots$ from left to right that operate synchronously. Each cell communicates with both left and right neighbors. The state of a cell c at time t is a function of the state of cells $c-1$, c, and $c+1$ at time $t-1$. The set of the states is finite and a subset of *accepting* states is distinguished. Here we consider states as signal emissions. Thus, the signal emission cell c at time t depends on signals received from the cells $c-1$, c, and $c+1$ at time $t-1$. Cell 1 communicates with the outside. The input mode is parallel: at time 0, the ith bit of the input string ω is on the ith cell; the remaining cells are in the quiescent state λ.

A useful way to present the evolution of a CA is a time-space diagram: the tth row corresponds to the configuration of the CA at time t, and the site (c, t) represents the cell c at time t. The information processing is expressed in terms of propagation and intersection of signals. These signals appear intuitively as an encoding of the information and its moves. In the time-space diagram they draw straight lines.

One-way cellular automata (OCA). An OCA is a CA where the neighborhood is restricted to the left cell: the evolution of a cell is defined by its own state and the state of its left neighbor.

Iterative arrays (IA). An IA has the same neighborhood as a CA. The restriction is on the input mode, which is sequential: the ith digit of the input is received by cell 1 at time i.

Language recognition. We say that a CA (IA or OCA) recognizes a language L in time $f(n)$ if it accepts words w of length n in $f(n)$ steps, that is, if cell 1 enters an accepting state at time $f(n)$. Real time corresponds to $f(n) = n - 1$; it is the minimal time required for cell 1 to read the input. Linear time corresponds to $f(n) = cn$ where c is a constant greater than 1. We write CA($f(n)$) (IA($f(n)$) or OCA($f(n)$) to denote the class of the languages accepted in time $f(n)$ by some CA (IA or OCA).

3. The language $\{0,1\}^* \cdot \{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$

Lemma 1. *The language* $\{0,1\}^* \cdot \{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$ *is recognizable by a CA in real time.*

Proof. In section 3.1 we use a process defined by Fisher [5] to organize the working area. In section 3.2 we define the moves of the input bits in order to prepare the comparison process, and in section 3.3 we describe the comparison process. ∎

3.1 Fisher's process

To define an IA that generates in real time the binary sequence representing the set of prime numbers, Fisher divides the working area in vertical slices of width $2, 3, \ldots, n$ cells and in each slice of width n sets up zigzagging signals to mark the multiples of n [5]. Here the slice of width n will test if the input word is of shape $\{0,1\}^* \cdot x^{|x|}$ with $|x| = n$. The zigzagging signals will be used to divide the input word into subwords of length n.

3.1.1 Dividing the working area into vertical slices of width $2, 3, \ldots, n$ (see Figure 1)

From the site $(1, 0)$ a signal R of slope 2 is sent. From the site $(2, 5)$ two signals are initialized: a vertical delay D, and a signal S of slope 1.

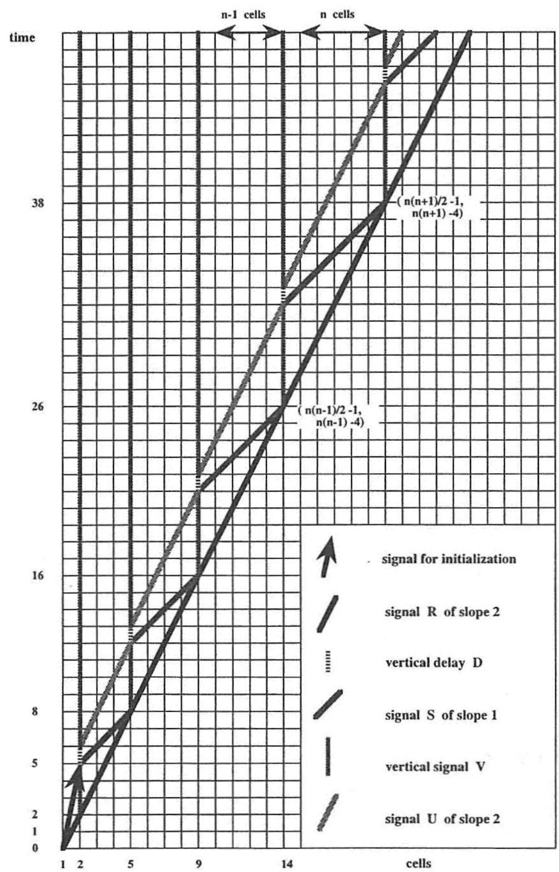

Figure 1: Dividing the working area.

The evolution is governed by the following rules.

1. When a site receives a vertical delay D, D stops and a signal U starts with slope 2.
2. At the intersection of a signal S and a signal R, S stops and R goes on its way; a vertical signal V is initialized.
3. At the intersection of a signal V and a signal U, V goes on its way and U stops; a vertical delay D and a signal S of slope 1 are initialized.

Observe that the signals S that reach signal R are characterized by the sites $(n(n+1)/2-1, n(n+1)-4)$. From each such site, a vertical signal V is initialized, denoted by V_n. Between V_{n-1} and V_n there are exactly n cells (excluding the cell V_{n-1}).

In the following, we keep only the signal R and the vertical signals V_n.

3.1.2 Setting up a sieve (see Figure 2)

In each slice we set up zigzagging signals where afterwards the bits will be collected. For $n > 2$, between the vertical signals V_{n-1} and V_n, from the site $(n(n+1)/2-1, n(n+1)-4)$ (where V_n is initialized), a signal is emitted to the left at speed 1. When it reaches the signal V_{n-1} it returns with a slope 2 to the signal V_n, where upon reaching V_n it runs again at speed 1 to V_{n-1}, thus bouncing between the two signals V_{n-1} and V_n.

In excluding the sites of V_{n-1}, this signal points out

- in its ith right move the set $R_{n,i}$ of the sites $(n(n-1)/2+k, n(n-1)-2+2k+3ni)$ with $k = 0, \ldots, n-1$
- in its ith left move the set $R_{n,-i}$ of the sites $(n(n-1)/2+k, n(n-1)-5-k+3ni)$ with $k = 0, \ldots, n-1$

For $n = 2$, a signal is emitted from site $(2, 2)$ (where V_2 is initialized) to the left at speed 1. When it reaches V_1, it remains 3 steps on V_1, and then returns with a slope 2 to signal V_2, and so on. This signal points out in its ith right move the set $R_{2,i} = \{(1, 6i), (2, 2+6i)\}$, and in its ith left move the set $R_{2,-i} = \{(1, -3+6i), (2, -4+6i)\}$.

3.2 Processing the bits of the input word

The purpose of this section is to prepare the data for the comparison process.

3.2.1 Moving the input word on the signal *R* (see Figure 2)

We assume that at the initial time the tth input bit x_t is on the cell $3t-2$; the remaining cells are in the quiescent state λ. First we move the input bits to the right at maximal speed to signal R. Thus the tth bit x_t emitted from site $(3t-2, 0)$ reaches the signal R at site $(t, 2t-2)$. More precisely, between V_{n-1} and V_n the n following bits reaching the set $R_{n,0}$ are $x_{n(n-1)/2+k}$ with $k = 0, \ldots, n-1$.

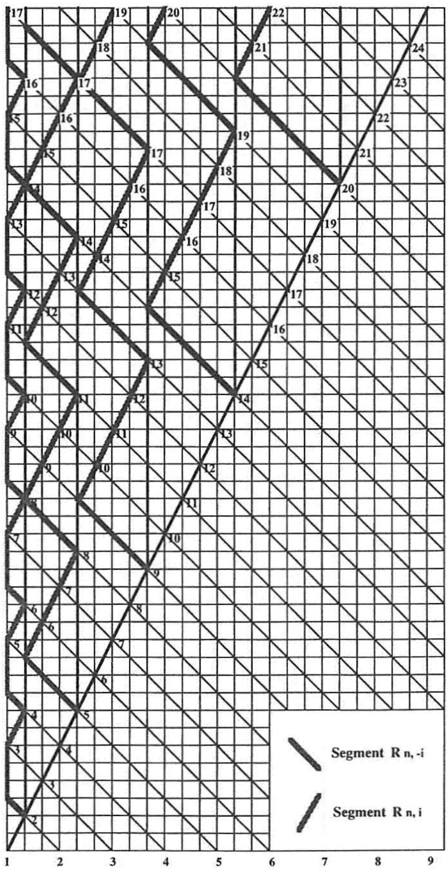

Figure 2: Left moves of the input bits.

3.2.2 Right moves of the bits (see Figure 3)

From signal R the bits proceed at maximal speed to the right. Observe that between V_{n-1} and V_n, the n bits reaching the set $R_{n,i}$ are $x_{n(n-1)/2+in+k}$ with $k = 0, \ldots, n-1$. More precisely, in the nth slice on the cell $c = n(n-1)/2+t$ (i.e., the tth cell of the nth slice), we get successively the bits $x_c, x_{c+n}, x_{c+2n}, \ldots, x_{c+in}, \ldots$.

3.2.3 Left moves of the bits

In order for cell $c = n(n-1)/2+t$ to obtain $x_{c-n}, x_{c-2n}, \ldots, x_{c-in}, \ldots$, the bits are dispatched in the following way. From the segment $R_{n,0}$, each bit is sent vertically to the first segment $R_{n,-1}$ and then runs to the right with a slope 2 until reaching a segment $R_{n+1,-i}$. From this segment, it runs vertically to

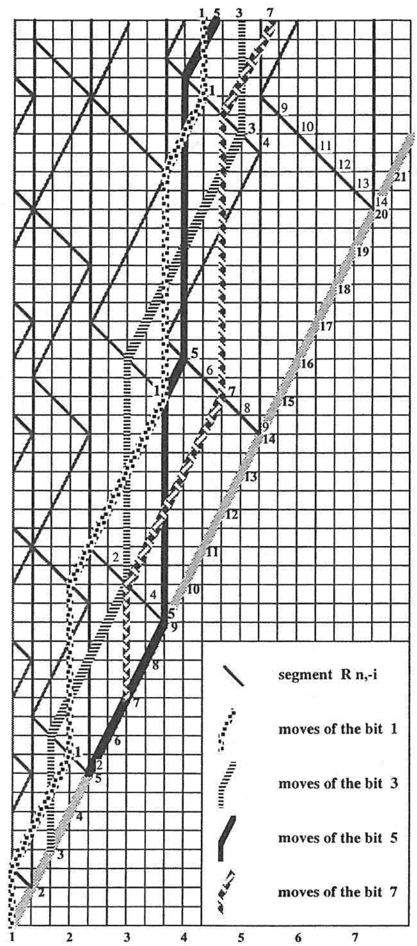

Figure 3: Right moves of the input bits.

the next segment $R_{n+1,-i-1}$, continuing to the right with a slope of 2, and so on. In this way, between V_{n-1} and V_n, the n bits reaching the segment $R_{n,-i}$ are $x_{n(n-1)/2-in+t}$ with $t = 0, \ldots, n-1$.

3.3 Comparison process

3.3.1 Notation

Let $|w|$ be the length of w and a, b be integers such that $|w|-(n+2)(n-1)/2 = an + b$ with $0 \leq b < n$. We denote by $r_{n,i}$ the subword $x_{n(n-1)/2-in} \cdots x_{n(n-1)/2-in+n-1}$, by $r_{n,i}[t]$ the bit $x_{n(n-1)/2-in+t}$, and by $r_{n,i}[s \ldots t]$ the subword $r_{n,i}[s]r_{n,i}[s+1] \ldots r_{n,i}[t]$.

After the previous section we observe that in the nth slice the word w is divided in this way: $w = dr_{n,1-n\text{div}2} \dots r_{n,0} \dots r_{n,i} \dots r_{n,a} r_{n,a+1}[1 \dots b]$ with $d = x_1 \dots x_{n-1}$ if n is odd, $x_1 \dots x_{n/2-1}$ if n is even, and $r_{n,a+1}[1 \dots b]$ the empty word if $b = 0$. Note that the beginning of the input word is obtained on the segment $R_{n,-n\text{div}2}$, which is below the segment $R_{n,a}$ if the length of the input word is at least n^2.

Observe that the word w is of shape $\{0,1\}^* \cdot x^{|x|}$ with $|x| = n$ if and only if the following three conditions are satisfied:

(c1) $r_{n,a} = r_{n,a-1} = \dots = r_{n,a-n+2}$;

(c2) $r_{n,a-n+1}[b+1 \dots n] = r_{n,a-n+2}[b+1 \dots n]$;

(c3) $r_{n,a+1}[1 \dots b] = r_{n,a}[1 \dots b]$.

3.3.2 Counting the number of successful comparisons

The past will be coded by means of two signals: a signal C that counts the number of successive identical subwords $r_{n,i}$, and a signal M that records the cell where the last failed comparison occurred. From segment $R_{n,-i}$, the counter C running on cell $c = (n+2)(n-1)/2 - T$ means that T previous subwords were identical: $r_{n,i} = r_{n,i-1} = \dots = r_{n,i-T+1}$. Actually, the counter C will be bounded by n, that is, C will not go through V_{n-1}. The signal M running on $c' = (n+2)(n-1)/2 - S$ implies that the last failed comparison occurred on cell c'; more precisely, in the case of $T < n$, it means that $r_{n,i} \neq r_{n,i-T}$, $r_{n,i}[n-S+1 \dots n] = r_{n,-T}[n-S+1 \dots n]$ and $r_{n,i}[n-S] \neq r_{n,i-T}[n-S]$.

The signals C and M are initialized on the segment R_{n-1}: on the cell $(n+2)(n-1)/2 - 1$ for the counter C on the cell $(n+2)(n-1)/2$ for the signal M. Between the two signals of type $R_{n,-i}$ they move vertically. We now show how these two signals evolve.

The n bits collected on segment $R_{n,i-1}$ are sent vertically to segment $R_{n,i}$ and compared two-by-two with the n bits collected on segment $R_{n,i}$. In the same way, the n bits collected on segment $R_{n,-i}$ are sent vertically and compared with the n bits collected on segment $R_{n,i}$.

The result of the two comparisons $r_{n,i} = r_{n,i-1}$ and $r_{n,i} = r_{n,-i}$ can be obtained on the right site of $R_{n,i}$. In addition, the right site of $R_{n,i}$ receives from the right site of $R_{n,i-1}$ a signal indicating that all the previous comparisons were successful.

Four cases can be distinguished (see Figure 4).

1. If $r_{n,i} = r_{n,i-1}$ and $r_{n,i} = r_{n,-i}$, and all the previous comparisons were successful, then the counter C is incremented by 2 and the signal M is not modified; in addition, the right site of $R_{n,i}$ sends to the right site of $R_{n,i+1}$ a signal indicating that all the previous comparisons were successful.

2. If $r_{n,i} = r_{n,i-1}$ and $r_{n,i} \neq r_{n,-i}$, and all the previous comparisons were successful, then the counter is incremented by 1 and the signal M is modified.

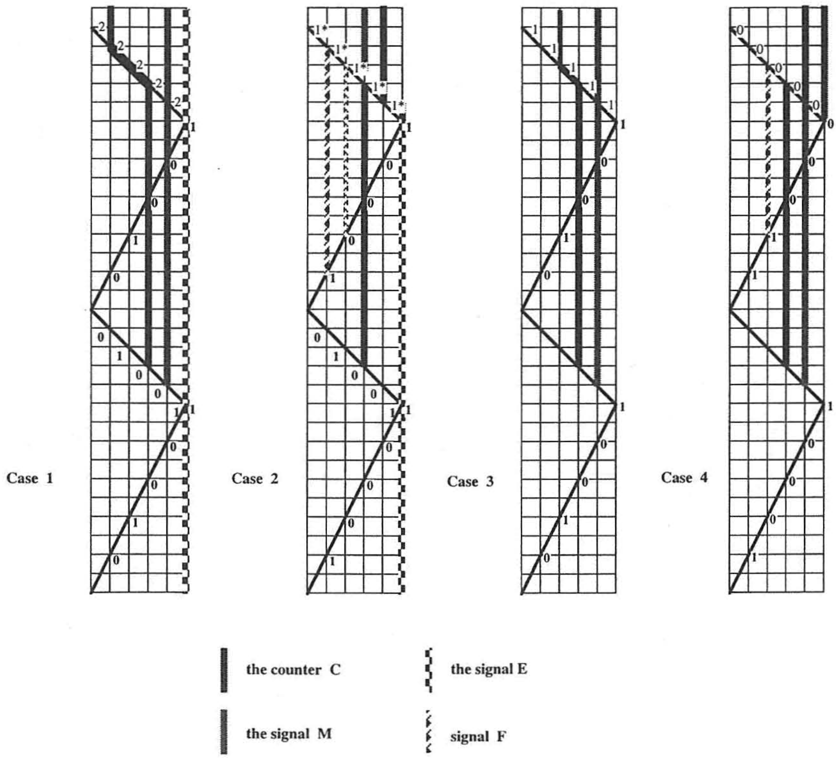

Figure 4:

3. If $r_{n,i} = r_{n,i-1}$ and one of the previous comparisons failed, then the counter is incremented by 1 and the signal M is not modified.

4. If $r_{n,i} \neq r_{n,i-1}$, then the counter is reinitialized and the signal M is modified.

So the right site of $R_{n,i}$ sends at maximal speed to the left along the segment $R_{n,-i-1}$ in case (1) a signal 2, in case (2) a signal 1*, in case (3) a signal 1, and in case (4) a signal 0.

A counter C receiving a signal 0 stops, and subsequently a new signal C is initialized on cell $(n+2)(n-1)/2-1$. A counter C receiving a signal 1 or 1* (respectively, a signal 2) moves one cell (respectively, two cells) to the left. When C reaches the signal V_{n-1}, it remains there (indeed the counter is bounded by n). A signal M receiving a signal 1 or 2 remains on the same cell, otherwise the signal stops and a new signal M is initialized on the cell where the last failed comparison occurred (on segment $R_{n,i}$ from the cells where a comparison failed, signals F are sent vertically to $R_{n,-i-1}$, on segment $R_{n,-i-1}$, the first signal F reached by signal 0 or 1* extends in signal M; the other signals stop).

3.3.3 Collecting the result

Now consider a signal E drawing by the move of the last bit $x_{|w|}$ of the input word. It starts initially from cell $3|w|-2$ and runs at maximal speed to the left. Observe that condition (c3) $r_{n,a+1}[1\dots b] = r_{n,a}[1\dots b]$ is satisfied if and only if the first b comparisons on segment $R_{n,a+1}$ are successful, that is, if and only if all comparisons until the intersection of the signal E and segment $R_{n,a+1}$ are successful. The result is obtained on the signal E.

In the other part, signal E intersects the counter C on the cell $(n+2)(n-1)/2-T$. Three cases can be distinguished according to the value of T.

- $T = n$. The counter C is on the vertical signal V_{n-1}, so the n previous comparisons are successful: $r_{n,a} = r_{n,a-1} = \dots = r_{n,a-n+2} = r_{n,a-n+1}$. Thus conditions (c1) and (c2) are both satisfied.

- $T < n-1$. The $n-1$ subwords $r_{n,a}, r_{n,a-1}, \dots, r_{n,a-n+2}$ are not all identical, so condition (c1) is not satisfied.

- $T = n-1$. The $n-1$ subwords $r_{n,a}, r_{n,a-1}, \dots, r_{n,a-n+2}$ are all identical, so condition (c1) is satisfied; in addition, if the signal E intersects the signal M on the cell $(n+2)(n-1)/2-S$, then we have $r_{n,i}[n-S+1\dots n] = r_{n,-T}[n-S+1\dots n]$ and $r_{n,i}[n-S] \neq r_{n,i-T}[n-S]$. Thus condition (c2) $r_{n,i}[b+1\dots n] = r_{n,-T}[b+1\dots n]$ is satisfied if and only if $S \geq n$, that is, if and only if the signal M intersects segment $R_{n,a+1}$ before signal E.

In conclusion, conditions (c1) and (c2) are satisfied if and only if signal E intersects the counter signal on V_{n-1} or signal E intersects the segment $R_{n,a+1}$ after the signal M and the counter signal one cell to the right of V_{n-1}. The result is obtained on the signal E. So if in a slice of length n the three conditions are satisfied, the signal E reaching the first cell at time $3|w|-3$ indicates to the first cell to enter an accepting state.

To conclude we consider the CA such that the cell (i,j) represents the cells $(3i+u, 3j+v)$ where $0 \leq u,v < 3$ of the previous CA. It recognizes $\{0,1\}^* \cdot \{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$ in real time.

Lemma 2. *The language $\{0,1\}^* \cdot \{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$ is not recognizable by an iterative array in real time.*

Proof. The proof of Cole shows that the language $\{0,1\}^* \cdot \{x$ is a palindrome: $x \in \{0,1\}^*, |x| > 2\}$ is not recognizable by an iterative array in real time ([3], Lemma 11, p. 362). The adaptation of this result to $\{0,1\}^* \cdot \{x^{|x|} : x \in \{0,1\}^*, |x| > 1\}$ is obvious. ∎

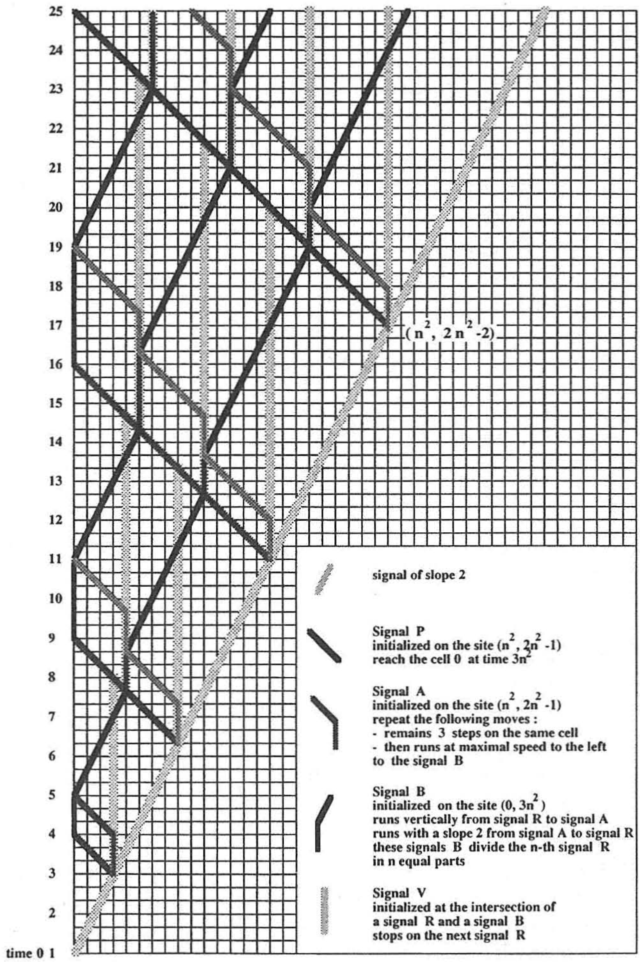

Figure 5:

4. The language $\{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$

Lemma 3. *The language* $\{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$ *is recognizable by an iterative array in real time.*

Proof. The algorithm is described in Figures 5 and 6.

As in section 3.1, we first characterize the sites $(n^2, 2n^2 - 2)$. From these sites, signals R are initialized, running at maximal speed to the left. They reach cell 1 at time $3n^2 - 3$. The area between the two signals R which reach cell 1 at times $3(n-1)^2 - 3$ and $3n^2 - 3$ will be devoted to the recognition of the words $x^{|x|}$ with $|x| = n$.

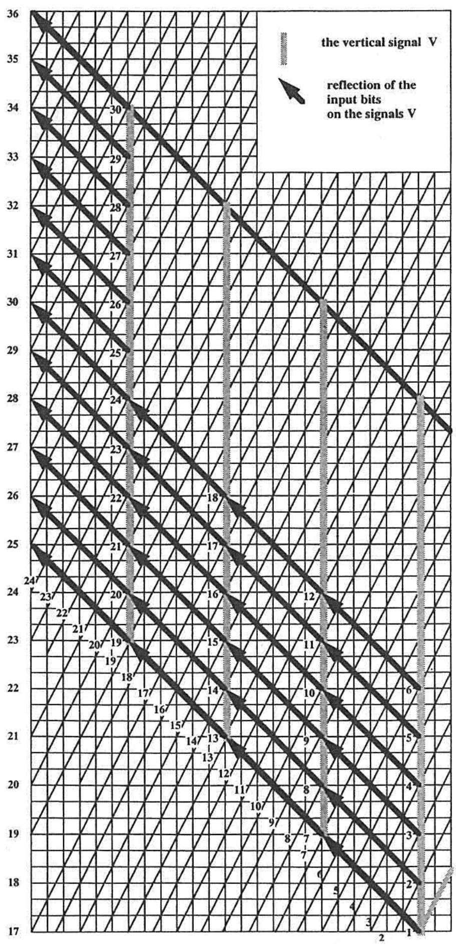

Figure 6:

Figure 5 organizes the work area by setting up vertical signals V which will distribute the input bits.

Figure 6 describes the moves of the input bits. We assume that the ith bit is fed to cell 1 at time $3i-3$ where it is sent with slope 2 to the right. In addition, when the bit x_i reaches a vertical signal V, it is sent to the right at maximal speed to the next vertical signal. So it is compared to the bit x_{i+n}, in the area devoted to the recognition of the words $x^{|x|}$ with $|x| = n$.

To conclude, we consider the IA such that the cell (i,j) represents the cells $(3i+u, 3j+v)$ where $0 \leq u, v < 3$ of the previous IA. It recognizes $\{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$ in real time. ∎

Lemma 4. *The language $\{x^{|x|} : x \in \{0,1\}^+, |x| > 1\}$ is not recognizable by a one-way cellular automaton in real time.*

Proof. Indeed, $\{1^{n^2} : n \in \mathbf{N}\}$ has been shown not to be a real-time OCA language [2] and OCA(n) is closed under intersection. Hence $L = \{x^{|x|} : x \in \{0,1\}^+, |x| > 1\} \notin \text{OCA}(n)$ since $L \cap \{1\}^* = \{1^{n^2} : n \in \mathbf{N}\}$.

References

[1] W. Bucher and K. Culik, "On Real Time and Linear Time Cellular Automata," *RAIRO Informatique Théorique*, **18**(4) (1984) 307–325.

[2] C. Choffrut and K. Culik II, "On Real-Time Cellular Automata and Trellis Automata," *Acta Informatica*, **21** (1984) 393–407.

[3] S. N. Cole, "Real-Time Computation by n-Dimensional Iterative Arrays of Finite-State Machine," *IEEE Transactions on Computers*, **C-18** (1969) 349–365.

[4] K. Culik, "Variations of the Firing Squad Problem and Applications," *Information Processing Letters*, **30** (1989) 153–157.

[5] P. C. Fischer, "Generation of Primes by a One-Dimensional Real-Time Iterative Array," *Journal of the ACM*, **12** (1965) 388–394.

[6] O. Ibarra and I. Jiang, "Relating the Power of Cellular Arrays to Their Closure Properties," *Theoretical Computer Science*, **57** (1988) 225–238.

[7] O. Ibarra and I. Jiang, "On One-Way Cellular Arrays," *SIAM Journal on Computing*, **16**(6) (1987) 1135–1154.

[8] A. R. Smith, "Real-Time Language Recognition by One-Dimensional Cellular Automata," *Journal of the ACM*, **6** (1972) 233–253.