Theoretical Computer Science 101 (1992) 59–98
Elsevier

A linear speed-up theorem for cellular automata

J. Mazoyer*
LIP, Ecole Normale Supérieure de Lyon, France

N. Reimen**
LITP, Institut Blaise Pascal, Université de Paris VII, Paris, France

Abstract

Mazoyer, J. and N. Reimen, A linear speed-up theorem for cellular automata, Theoretical Computer Science 101 (1992) 59–98.

Ibarra et al. (1985) showed that, given a cellular automaton of range 1 recognizing some language in time $n+1+R(n)$, we can obtain another CA of range 1 recognizing exactly the same language but in time $n+1+R(n)/k$ ($k \geqslant 2$ arbitrary). Their proof proceeds indirectly (through the simulation of CAs by a special kind of sequential machines, the STMs) and we think it misses that way some of the deep intuition of the problem. We, therefore, provide here a direct proof of this result (extended to the case of CAs of arbitrary range) involving the explicit construction of a CA working in time $n+1+R(n)/k$. This speeded-up automaton first gathers the cells of the line k by k in $n+1$ steps which then enables it to start computing by "leaps" of k steps, thus completing the $R(n)$ remaining steps in time $R(n)/k$. The major problem arising from the obligation to pass from one phase to the other synchronously is solved using a synchronization process derived from the solutions of the well-known "firing-squad synchronization problem" (FSSP).

1. Introduction

The idea of using cellular automata for computation dates back to their origin, since Von Neumann's self-reproducing patterns [13] were also capable of universal computation. This ability relied on the embedding of a simulated Turing machine, but it was soon discovered that the use of CAs for their own sake allowed very fast computation, strictly faster, indeed, than with TMs. The multiplication of two integers [1] and primality testing [5] were thus shown to be computable by a one-dimensional

*Supported by PRC Mathématiques et Informatique and Esprit Basic Research Action Working Group: Algebraic and Syntactical Methods in Computer Science (ASMICS).

**Supported by PRC Mathématiques et Informatique. This work was conducted while the author was a student at the Ecole Normale Supérieure de Paris.

CA in real time. Smith [11] obtained the same result for a large sample of language recognition problems, among which some were not real-time TM-computable, thus proving the inherent superiority of CAs.

As soon as CAs were used as computing devices, it was natural to attempt to adapt to them the complexity-theoretic theorems known for TMs and, among them, the most basic, the linear speed-up theorem. Cole [3] and Ibarra et al. [6] proved two versions of this theorem in the case of CAs taken as language recognizers. The proof of such a result, as in the case of TMs, involves the working out of an automaton k times faster, for some k, than another one previously given, i.e. one which would do in one step what the latter did in k. Each cell of the accelerated automaton must, therefore, be provided with k times more information than one of the original automaton.

Cole [3] achieved this by grouping the cells k by k, as in the usual proof of the theorem for TMs, obtaining a recognition time of $\lceil t(n)/k \rceil$, where $t(n)$ is the recognition time of the initial automaton. But he required the grouping to be effected "by the user" so as to provide the accelerated machine with an initial configuration already grouped. In this way he obtained what we will refer to as the "weak form" of the linear speed-up theorem for CAs, allowing the recognition not exactly of the initial language but of that obtained by grouping the letters of its words k by k. On the contrary, the "strong form" corresponds to the ability to recognize the initial language exactly. One way to obtain this strong form is through an increase by a factor k of the range[1] of the transition function, thus providing the same improvement in information availability without requiring any modification of the alphabet and, hence, of the language recognized. However, this is not completely satisfactory, since there are many cases where the range is precisely required to remain fixed notably when only CAs of range 1 are desired (all papers mentioned here deal with this common kind of CA).

Ibarra et al. [6] proved the strong form in the case where all automata considered are of range 1, but they did this through indirect arguments. They showed that CAs of range 1 are equivalent, in time, to a special class of sequential machines, the STMs (sweep Turing machines), which proceed through a sequence of "sweeps", i.e. of passes of the read/write head from the left to the right of the nonblank part of the tape, each counted as one time step. They subsequently proved a speed-up theorem in strong form for STMs yielding a recognition time of $n+1+\lceil [t(n)-n-1]/k \rceil$ and, thus, obtained the same result for CAs of range 1.

The main result presented in this paper is a direct proof of the result of Ibarra et al. extended to the case of CAs of arbitrary range. It is direct in the sense that it involves the explicit construction of a cellular automaton fulfilling the time requirements of the theorem. To do this we develop a general composition technique for cellular automata involving as its key component a synchronization process derived from the solutions of the FSSP (see e.g. [9, 7]). In addition, we obtain new results for the variant of the FSSP in which the synchronization process begins at both ends of the segment (in the

[1] In a cellular automaton, the state of each cell at time $t+1$ is determined as a function of the state at time t of itself and of that of its neighbours lying within range p on both sides. This number p is called the *range* of the cellular automaton.

classical FSSP, it only begins at the left end). We show here that in that case the minimal synchronization time is τ for a segment of length τ (it is $2\tau-1$ in the case of the classical FSSP) and provide a solution which works in minimal time. We use this synchronization process to compose the automaton corresponding to the weak form of the theorem—which we restate in passing—with a grouping automaton which enables the resulting device to be fed with ungrouped data. This approach shows that our result may be obtained without resorting to anything outside the cellular automata theory, thus preserving its "self-containedness". Moreover, our construction is much more efficient in terms of the number of states of the speeded-up automaton than that of Ibarra et al. from a CA of range 1 with s states, we obtain a speeded-up automaton with $O(s^k)$ states, where k is the speed-up factor, whereas the one obtained through the method of Ibarra et al. has $O(s^{12k})$ states. Observe that $O(s^k)$ is most probably optimal. In addition, we feel that providing solutions stated in cellular-automata-theoretic terms allows a better understanding of the problems involved. We were also able to take advantage of this opportunity to develop several useful techniques of cellular automata construction which are interesting in themselves.

2. Definitions

A *one-dimensional cellular automaton* (CA) of *range* p is a TM-ribbon-like structure, that is a bi-infinite line of squares (called here *cells*) each containing a symbol taken in a finite alphabet Q, together with a *local transition function* $f: Q^{2p+1} \to Q$. Performing one *step* of the automaton is achieved by computing a new content for each cell, using f, as a function of the value of the neighbouring cells lying within range p on both sides. $A=(Q,p,f)$ will denote a CA with these attributes. A *configuration* of A is an element of $Q^{\mathbf{Z}}$ which maps the integers (understood as cell numbers) to the contents of the cells at a given time. On $Q^{\mathbf{Z}}$, the set of all possible configurations, f induces a *global map* $G_f: Q^{\mathbf{Z}} \to Q^{\mathbf{Z}}$ such that for all c in $Q^{\mathbf{Z}}$, if $c'=G_f(c)$, then for all $i \in \mathbf{Z}$ we have

$$c'(i)=f(c(i-p),\ c(i-p+1),\ldots,c(i-1),\ c(i),\ c(i+1),\ldots,c(i+p)).$$

A *recognizer CA* is a structure $\mathcal{A}=(U,\lambda,q_0,Q_a,Q_r,A)$ where $A=(Q,p,f)$ is a 1-d CA (henceforth we shall simply say *CA* for *one-dimensional CA*). As is usual, when using CAs as computing machines, we shall use semi-infinite lines instead of the bi-infinite ones defined above. The reasons for such a choice being numerous and somewhat vague, it would take too much space to give them here. If we were to adhere strictly to this semi-infinitism, we would have to consider two classes of cells: "normal cells" and "end cells", those close to the finite end of the line, which would have a different number of neighbours and, hence, whose behaviour would have to be handled by different transition functions. Nevertheless, to be able to use the formalism developed above, we shall replace semi-infinite lines by bi-infinite ones with all negative cells "forbidden" by a special *border state*, $\lambda \in Q$. A cell once in state λ will remain in that state forever, i.e. $f(x_{-p},\ldots,x_{-1},\lambda,x_1,\ldots,x_p)=\lambda$, whatever the x_i. We

also require that no positive cell will ever be in state λ. One single transition function is thus sufficient to describe the behaviour of all cells, the "end cells" being those with at least one λ lying within range p on the left. The leftmost cell, bearing number 0, is called the *distinguished cell*. Another special state is the *quiescent state*, $q_0 \in Q$, with the following property: $f(q_0, q_0, \ldots, q_0) = q_0$. It plays a role similar to that of the blank symbol of TMs. U, a subset of Q which must not contain λ or q_0, is called the *input alphabet* of $\mathscr{A}$. To any input word $u = u_0 u_1 \ldots u_{n-1} \in U^*$ corresponds an *initial configuration*, denoted $c_{\mathscr{A}}[u]$, which assigns to negative cells the state λ, to cell i, for $i \in \{0, 1, \ldots, |u|-1\}$, the letter u_i and to all the following cells the state q_0. We have

$$c_{\mathscr{A}}[u] = \ldots \lambda\lambda u_0 u_1 \ldots u_{n-1} q_0 q_0 \ldots,$$

with u_0 in position 0. The computation of $\mathscr{A}$ on u yields the following sequence of configurations[2]

$$c_{\mathscr{A}}[u],\ G_f(c_{\mathscr{A}}[u]),\ G_f^{(2)}(c_{\mathscr{A}}[u]), \ldots, G_f^{(x)}(c_{\mathscr{A}}[u]), \ldots$$

Q_a and Q_r are, respectively, the sets of *accepting* and *rejecting* states. They are disjoint subsets of Q and must not have any intersection with $U \cup \{q_0, \lambda\}$. We shall say that $\mathscr{A}$ *recognizes* language $L \subset U^*$ in time $t(n)$ if, for a word $u \in L$ ($\notin L$), the distinguished cell (cell 0) enters an accepting (rejecting) state at time $t(|u|)$, i.e. if

$$[G_f^{t(|u|)}(c_{\mathscr{A}}[u])](0) \in Q_a (Q_r).$$

To deal with a sequence of configurations $(c_t)_{t \in \mathbf{N}}$ such as those produced by the execution of a CA, we shall often use the application of $\mathbf{N} \times \mathbf{Z} \to Q$ induced by (c_t) which maps the pairs (t, k) to the contents of cell k at time t, i.e. $c_t(k)$. The pairs (t, k) will be called the *sites* of the *space–time diagram* (STD) associated to the sequence (c_t). If we denote by $\langle t, k \rangle$ the state of the site (t, k) of the diagram associated with the sequence (c_t), we have the relation $\langle t, k \rangle = c_t(k)$, which shows that the space–time diagram is simply the decurryfied[3] form of (c_t). We shall henceforth say "the site $\langle t, k \rangle$" instead of "the site (t, k) of the diagram associated to the sequence (c_t)". If the sequence (c_t) is related to a CA of range p, we may define a notion of "visibility" on the sites of the STD associated with it. We shall say that site $\langle t, k \rangle$ *sees* site $\langle t-1, k' \rangle$ if, to compute the former, a transition function of range p takes the latter into account, that is to say if $|k - k'| \leqslant p$. We use this notion extensively when, given a certain behaviour expressed in terms of a sequence of configurations or a STD, we want to prove that there is a CA with such a behaviour and many of our proofs proceed more or less.

It is natural, and indeed usual, to translate this notion of STD graphically. Sometimes, sites will be seen as dots linked by lines dividing the space–time plane into

[2] We write $G_f^{(k)}$ for $\underbrace{G_f \circ G_f \circ \cdots \circ G_f}_{k}$.

[3] *If* $f: A \times B \to C$ then the curryfied form of f is the function $\mathrm{cur}f: A \to (B \to C)$, such that $[(\mathrm{cur}f)(x)](y) = f(x, y)$.

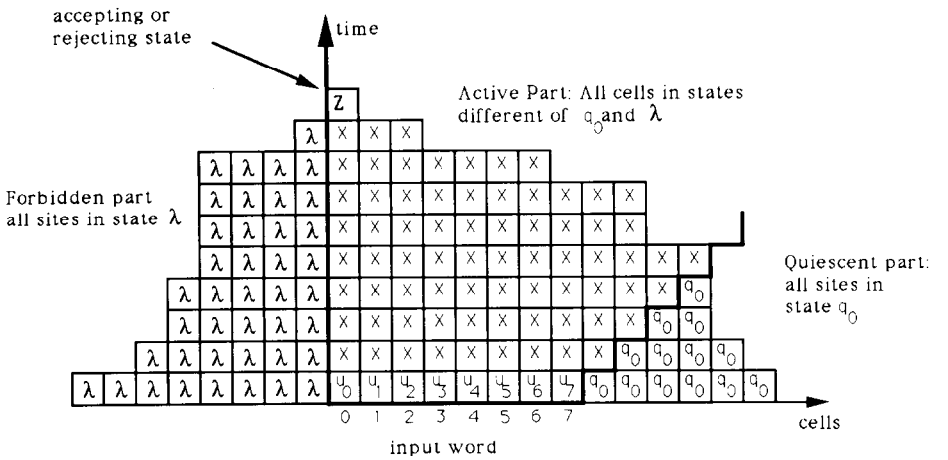

Fig. 1.

zones of common behaviour, thus giving a "continuous approximation" of the discrete structure of STDs. This enables us to give a rough but meaningful representation of the global features of the diagrams studied. As an example, Fig. 1 shows the structure of the STD of a recognizer CA. However, when a more detailed picture is sought, it is convenient to come back to a discrete space and display the content of each individual cell as, for example, in Fig. 5.

3. Doing k steps in one

We show here that, given a cellular automaton and a constant $k \leqslant 2$, we can build another CA which performs in one step, on grouped configurations, what the initial one did in k on the corresponding ungrouped configurations. From this lemma one can obtain, with little extra work, the automaton fulfilling the requirements of the weak form of the speed-up theorem and that will, in turn, be used in the proof of the strong form.

In the following, $A=(Q, p, f)$ will denote the initial automaton and $B=(R, q, g)$ the automaton we intend to build with in all cases $R=Q^{\alpha}$ for a given integer $\alpha \geqslant 2$. This means that we want each cell of B to contain α cells of A. On the other hand, we want B to do in one step what A does in k. If c and $c'=G_g(c)$ are two consecutive configurations of B, we want d', the ungrouped configuration corresponding to c', to be obtained from d, the ungrouped configuration corresponding to c, through k steps of the transition function of A. We want

$$d'=G_f^{(k)}(d).$$

Let us say that d' corresponds to the instant t and d to the instant $t-k$. Computing the state of one cell of c' means computing the state of α consecutive cells of A at time t. To

compute them, it is necessary to know the states of our α cells and those of the neighbouring cells lying within range p on both sides at time $t-1$. It is then, in turn, necessary, in order to compute the states of these $2p+\alpha$ cells at time $t-1$, to know their states plus again those of their p neighbours on each side at time $t-2$, and so on ... If we go back until time $t-k$, we see that we have to know, at that moment, the states of kp cells on each side of the α initial cells (see Fig. 2). To compute the states of α consecutive cells in d', we must therefore know their states in d and those of kp cells of d on each side. Our α cells of d' corresponds to one cell of c' whose value is determined by g, the transition function of B, according to its state in c and the states of q cells of c on each side. These q cells contain $q\alpha$ cells of d. Since we need at least kp of them, we must have the following relation:

$$q\alpha \geqslant kp.$$

We shall now formalize the idea developed above.

Let γ_α be the bijective function from $Q^{\mathbf{Z}}$ to $(Q^\alpha)^{\mathbf{Z}}$, which associates to the ungrouped configuration c on Q the corresponding grouped configuration $\gamma_\alpha(c)$ on Q^α. We have

$$[\gamma_\alpha(c)](i)=(c(\alpha i), c(\alpha i+1),\ldots,c((\alpha+1)i-1)).$$

We shall use it to go back and forth between the configurations of A and those of B.

Lemma 3.1. *Let $A=(Q,p,f)$ be a cellular automaton. If α, q and k are three nonnegative integers such that $q\alpha \geqslant kp$ then there exists another CA $B=(Q^\alpha,q,g)$ such that $\gamma_\alpha^{-1}\circ G_g\circ\gamma_\alpha=G_f^{(k)}$ (B does in one transition what A does in k).*

In order to prove this lemma we first need a preliminary result.

Proposition 3.2. *Let c_1 and c_2 be two configurations on Q which are equal on all cells i such that $-q\alpha \leqslant i \leqslant q\alpha+\alpha-1$. If $q\alpha \geqslant kp$ then the configurations $G_f^{(k)}(c_1)$ and $G_f^{(k)}(c_2)$ are equal on all cells j such that $0 \leqslant j \leqslant \alpha-1$.*

Proof. $G_f(c_1)$ and $G_f(c_2)$ are equal on all cells i such that $-q\alpha+p \leqslant i \leqslant (q+1)\alpha-1-p$ (see Fig. 3). So $G_f^{(k)}(c_1)$ and $G_f^{(k)}(c_2)$ are equal on all cells j such that $-q\alpha+kp \leqslant$

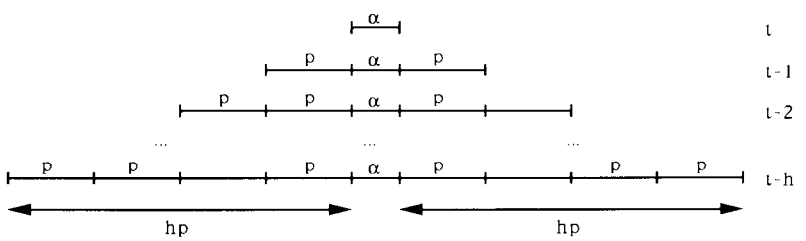

Fig. 2.

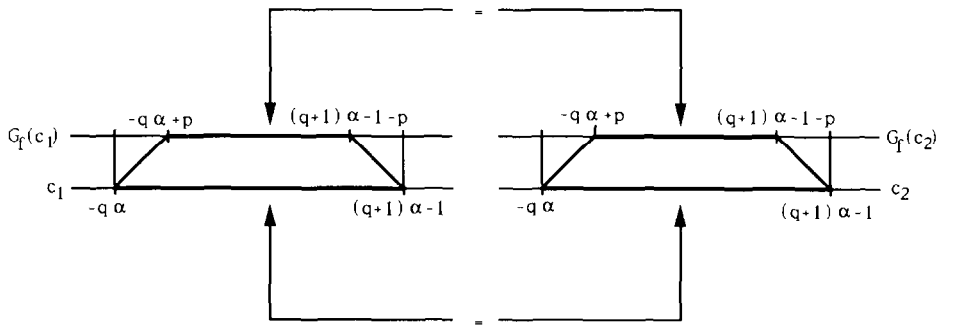

Fig. 3.

$j \leqslant (q+1)\alpha - 1 - kp$. Since $q\alpha \geqslant kp$, we have $-q\alpha + kp \leqslant 0$ and $(q+1)\alpha - 1 - kp \geqslant \alpha - 1$. So, $G_f^{(k)}(c_1)$ and $G_f^{(k)}(c_2)$ are equal on all cells j such that $0 \leqslant j \leqslant \alpha$. □

We have thus shown that, if $q\alpha \geqslant kp$ holds, then the states of cells 0 to $\alpha - 1$ (and, hence, of any group of α consecutive cells) at time t may be determined if we know their states and those of q consecutive groups of α cells on each side at time $t-k$. We may now give a proof of Lemma 3.1.

Proof of Lemma 3.1. Let us assume that the condition $q\alpha \geqslant kp$ is fulfilled. We shall define B's transition function in the following way. Let

$$v = (w_{-q}, \ldots, w_{-1}, w_0, w_1, \ldots, w_q)$$

be an element of $(Q^\alpha)^{2q+1}$ of which we want to determine the image by g. Let c_v be an arbitrary configuration on Q^α such that $c_v(i) = w_i$ for all i such that $-q \leqslant i \leqslant q$. We let

$$g(v) = [\gamma_\alpha(G_f^{(k)}(\gamma_\alpha^{-1}(c_v)))](0).$$

Figure 4 gives a more readable interpretation of the above formula. Due to the way in which we defined c_v, we know that the choice of one v assigns a value to all cells $-q\alpha$

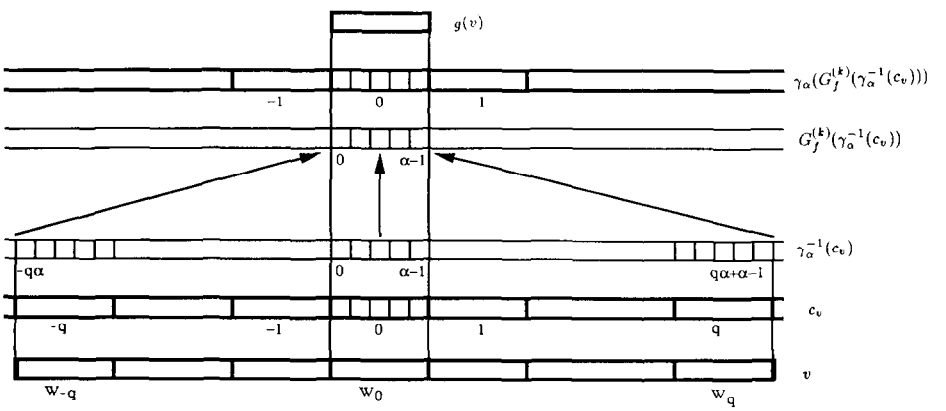

Fig. 4.

to $q\alpha+\alpha-1$ of $\gamma_\alpha^{-1}(c_v)$. In these conditions, Proposition 3.2 tells us that the states of the cells 0 to $\alpha-1$ of $G_f^{(k)}(\gamma_\alpha^{-1}(c_v))$ are uniquely determined since we have $q\alpha \geqslant kp$. We are, therefore, allowed to define g that way and it is clear that we have

$$G_g=\gamma_\alpha \circ G_f^{(k)} \circ \gamma_\alpha^{-1}. \qquad \square$$

4. The speed-up theorem—weak form

Let $\mathscr{A}=(U, q_0, \lambda, Q_a, Q_r, A)$ with $A=(Q, p, f)$ be a recognizer CA which recognizes the language $L \subset U^*$ in time $t(n)$. From A and for any $\alpha \geqslant 2$, $q \geqslant 1$ and $k \geqslant 1$ such that $q\alpha \geqslant kp$, Lemma 3.1 provides an automaton $B=(Q^\alpha, q, g)$, whose transition function does in one step on a grouped configuration what f does in k on the corresponding ungrouped configuration. We shall build from B another recognizer CA $\mathscr{B}$ which will start from grouped configurations and will tell, within time $\lceil t(n)/k \rceil$, if the word of length n contained in the initial grouped configuration belongs to L or not. For a word $u=u_0u_1 \ldots u_{n-1} \in U^*$ the initial configuration for $\mathscr{A}$ is a configuration $c_{\mathscr{A}}[u]$ of the form

$$\ldots \lambda\lambda\lambda u_0 u_1 u_2 \ldots u_{n-2} u_{n-1} q_0 q_0 q_0 \ldots,$$

with u_0 in position 0. The corresponding initial configuration for $\mathscr{B}$ will be $\gamma_\alpha(c_{\mathscr{A}}[u])$ (we assume for this example that n is not a multiple of α):

$$\ldots(\underbrace{\lambda, \ldots, \lambda}_{\alpha})(\underbrace{\lambda, \ldots, \lambda}_{\alpha})(u_0, \ldots, u_{\alpha-1})(u_\alpha, \ldots, u_{2\alpha-1}) \ldots$$

$$\ldots(u_{\lfloor n/\alpha \rfloor \alpha}, \ldots, u_{n-1}, \underbrace{q_0, \ldots, q_0}_{\alpha-n \bmod \alpha})(\underbrace{q_0, \ldots, q_0}_{\alpha}) \ldots,$$

with $(u_0, \ldots, u_{\alpha-1})$ in position 0. This configuration must be, according to the definition of recognizer CAs, a $c_{\mathscr{B}}[v]$, where v depends on u. To do this, we shall let[4] $(\lambda, \ldots, \lambda)_\alpha$ be the border state of $\mathscr{B}$ and $(q_0, \ldots, q_0)_\alpha$ its quiescent state. We shall also generalize γ_α to the words and languages in the following way: Let u be a word of U^*, $\gamma_\alpha(u)$ is the word on $((U \cup \{q_0\})^\alpha)^*$ obtained by grouping α by α the letters of u and by padding with q_0 the space left in the last letter if $|u|$ is not a multiple of α. With the u of the above example we obtain

$$\gamma_\alpha(u)=(u_0, \ldots, u_{\alpha-1}) \ldots (u_{\lfloor n/\alpha \rfloor \alpha}, \ldots, u_{n-1}, \underbrace{q_0, \ldots, q_0}_{\alpha-n \bmod \alpha}).$$

This way we have $v=\gamma_\alpha(u)$ and, thus, $c_{\mathscr{B}}[\gamma_\alpha(u)]=\gamma_\alpha(c_{\mathscr{A}}[u])$. The generalization of γ_α to languages is straightforward: $\gamma_\alpha(L)=\{\gamma_\alpha(u), u \in L\}$. The language recognized by the

[4] We write $(x, y, \ldots, z)_\alpha$ for $\underbrace{(x, y, \ldots, z)}_{\alpha}$

$\mathscr{B}$ we shall build will, therefore, be $\gamma_\alpha(L)$. The "weak" aspect of the version of the theorem we develop here comes from the fact that it is $\gamma_\alpha(L)$ and not L which is recognized by B.

Theorem 4.1. *Let $\mathscr{A}=(U, q_0, \lambda, Q_a, Q_r, A)$ with $A=(Q, p, f)$ be a recognizer CA which recognizes the language $L \subset U^*$ in time $t(n)$. Let α, q, k be three positive integers such that $q\alpha \geqslant kp$. There exists a recognizer CA $\mathscr{B}=(((U \cup \{q_0\})^\alpha)^*, (q_0, \ldots, q_0)_\alpha, (\lambda, \ldots, \lambda)_\alpha, R_a, R_r, B^*)$ with $B^*=(Q^\alpha, q, g^*)$ which recognizes the language $\gamma_\alpha(L)$ in time $\lceil t(n)/k \rceil$.*

Proof. If we take $B^*=B$, where B is the automaton whose existence is given by Lemma 3.1, $R_a=Q_a \times Q^{\alpha-1}$ and $R_r=Q_r \times Q^{\alpha-1}$, we obtain the behaviour desired, but only if $t(n)$ is a multiple of k, in which case the configuration

$$G_g^{(t(n)/k)}(\gamma_\alpha(c_{\mathscr{A}}[u]))$$

corresponds to

$$G_f^{(t(n))}(c_{\mathscr{A}}[u])$$

and, therefore, contains the end state (accepting or rejecting) in cell 0. But if $t(n)$ is not a multiple of k then the end state will appear *between* two of the configurations computed by B and, therefore, be lost. To avoid this problem we shall modify g^*, the transition function of B^*, so that if an end state appears in cell 0 (if it appears in another cell we do not care since end states "count" only if they appear in cell 0) at time $t_1 k + t_2$ in the computation of $\mathscr{A}$ with $t_1 \geqslant 0$ and $1 \leqslant t_2 \leqslant k-1$ then it also appears as the rightmost letter of the α-uple

$$[G_{g^*}^{(t_1+1)}(\gamma_\alpha(c_{\mathscr{A}}[u]))](0),$$

that is the state of the cell 0 at time t_1+1 in the computation of B^*. This means that if an end cell appears between two instants of B^* then it is "saved" by g^* and, thus, appears at the next instant of B^*. We showed in the proof of Lemma 3.1 that if the relation $q\alpha \geqslant kp$ holds then g can obtain from the configuration of B at time t_1 enough information about the configuration of $\mathscr{A}$ at time $t_1\alpha$ to compute the state at time $(t_1+1)\alpha$ of the α cells of $\mathscr{A}$ contained in one cell of B at time t_1+1. It has, therefore, also enough information to compute the states of these α cells of $\mathscr{A}$ at time $t_1\alpha+t_2$ for any $t_2 \in \{1, \ldots, k-1\}$ and check if the leftmost of them is an end state. g^* may also check if it is dealing with cell 0 by checking if its left neighbour contains a λ state. g^* may, therefore, be arranged so as to yield an α-uple with the appropriate end state (found at time $t_1\alpha+t_2$) as its leftmost letter if these two conditions are fulfilled. Keeping $R_r=Q_r \times Q^{\alpha-1}$ and $R_a=Q_a \times Q^{\alpha-1}$ we obtain the desired result. □

Corollary 4.2. *If the language L is recognized by a recognizer CA of range p in time $t(n)$ then L is also recognized by another recognizer CA of range kp in time $\lceil t(n)/k \rceil$.*

Proof. To enable the recognition of L there must not be any grouping; so, $\alpha = 1$. If we take $q = kp$, we have obviously $q\alpha \geqslant kp$; so, Theorem 4.1 provides us with a CA which fulfils the requirements of the corollary. □

This way we obtain the strong form of the speed-up theorem but at the price of a significant increase in the range. We explained in the introduction why we were not ready to pay such a price.

Corollary 4.3. *If the language L is recognized in time $t(n)$ by a recognizer CA of range p then $\gamma_{kp}(L)$ is recognized by a recognizer CA of range 1 in time $\lceil t(n)/q \rceil$.*

This shows that we may accelerate the computations of automata of arbitrary range by only relying on automata of range 1. However, in that case, the price to pay is the inflation of the number of states.

5. Grouping

To be able to prove the strong form of the speed-up theorem using the ideas developed so far, we need an "adapter" to turn the initial ungrouped configuration into a grouped one from which the accelerated computation may start. Our starting point is a recognizer CA $\mathcal{A} = (U, \lambda, q_0, Q_a, Q_r, A)$, with $A = (Q, p, f)$ which recognizes a certain language $L \subset U^*$ in time $t(n)$. We intend to build another recognizer CA $\mathcal{C}$ which recognizes the same language in accelerated time. We shall achieve speed-up in $\mathcal{C}$ through re-use of the automaton B^* of the weak form. Since we do not want to impose an increase in the range, we must rely on grouping to obtain speed-up. So, B^* will need to work on grouped configurations. But the initial configuration of $\mathcal{C}$ cannot be fully grouped since the input word u must not be such as to enable $\mathcal{C}$ to recognize L and not $\gamma_\alpha(L)$. This is why we need a grouping adapter. We shall take the border state of $\mathcal{C}$ equal to[5] $(\lambda, \ldots, \lambda)_\alpha$ and its quiescent state equal to $(q_0, q_0, \ldots, q_0)_\alpha$ (for some $\alpha \geqslant 2$). This way, the initial configuration[6] of $\mathcal{C}$ for a word $u = u_0 u_1 \ldots u_{n-1} \in U^*$ is

$$c_{\mathcal{C}}[u] = \ldots \underbrace{(\lambda, \ldots, \lambda)}_{\alpha} \underbrace{(\lambda, \ldots, \lambda)}_{\alpha} u_0 u_1 u_2 \ldots u_{n-1} \underbrace{(q_0, \ldots, q_0)}_{\alpha} \underbrace{(q_0, \ldots, q_0)}_{\alpha} \ldots,$$

with u_0 assigned to cell 0. This configuration is "almost grouped" since only a finite number of its positions are not. We shall build a grouping automaton B^G which will turn $c_{\mathcal{C}}[u]$ into a fully grouped configuration (i.e. the image by γ_α of a configuration of $\mathcal{A}$) from which the accelerated computation of B^* may start. $\mathcal{C}$ will then be the result of the composition of B^G and B^*.

[5] We write $(x, x, \ldots, x)_\alpha$ for $\underbrace{(x, x, \ldots, x)}_{\alpha}$.

[6] We recall here that the initial configuration of a recognizer CA for a word u assigns a border state to the negative cells, and to the positive cells the letters of u followed by the quiescent states.

To formalize this notion of "almost grouped configurations" we introduce the family of functions $\gamma_\alpha^k : Q^{\mathbf{Z}} \to (Q^\alpha \cup Q)^{\mathbf{Z}}$ for any $k \in \mathbf{N}$. Intuitively, the configuration $\gamma_\alpha^k(c)$ is that obtained by grouping the configuration c everywhere except on cells 0 to $k-1$. Formally, for any configuration $c \in Q^{\mathbf{Z}}$ we have

$$[\gamma_\alpha^k(c)](i) = \begin{cases} c(i) & \text{if } 0 \leqslant i < k, \\ (c(I), c(I+1), \ldots, c(I+\alpha-1)), & \\ \text{with } I = k+(i-k)\alpha & \text{if } i \geqslant k, \\ (c(i\alpha), c(i\alpha+1), \ldots, c(i\alpha+\alpha-1)) & \text{if } i < 0. \end{cases} \tag{1}$$

Thus defined, the functions γ_α^k are also bijective, like γ_α, and we have $\gamma_\alpha^0 = \gamma_\alpha$. $c_{\mathscr{C}}[u]$, the initial configuration for $\mathscr{C}$, is equal to $\gamma_\alpha^n(c_{\mathscr{A}}[u])$. We shall say that a configuration is γ_α^k-*grouped* if it may be obtained through γ_α^k from a configuration on Q, in that sense, $c_{\mathscr{C}}[u]$ is γ_α^n-grouped.

For technical reasons, which will appear soon, we cannot obtain B^{G} directly. We show here first that there exists an automaton B^{g}, of range $q \geqslant p$, (p is the range of $\mathscr{A}$) which, starting from the configuration $c_{\mathscr{C}}[u]$, yields a $\gamma_\alpha^{n \bmod q}$-grouped configuration at time $\lfloor n/q \rfloor$. To obtain the final γ_α^0-grouped configuration of B^{G} we shall have to compute the value of $n \bmod q$, and communicate it to all cells 0 to $n+q-n \bmod q-1$ (all nonquiescent cells of the final configuration of B^{G}; see the end of this section) and to synchronize all these cells at time $\lfloor n/q \rfloor + 1$ (see Section 7). All this being done, we shall finally build B^{G} in Section 8.

Let us now begin the construction of B^{g}. We are given a recognizer CA $\mathscr{A}$ which, fed with the initial configuration $c_{\mathscr{A}}[u]$ corresponding to a word $u = u_0 u_1 \ldots u_{n-1}$, yields the sequence of configurations

$$c_t = G_f^{(t)}(c_{\mathscr{A}}[u])$$

for all $t \geqslant 1$. To this sequence of configurations is associated a space–time diagram the sites of which we shall denote by $\langle t, k \rangle$ for $t \in \mathbf{N}$ and $k \in \mathbf{Z}$. For $t = 0$ we have $\langle 0, k \rangle = \lambda$ if $k < 0$, $\langle 0, i \rangle = u_i$ if $0 \leqslant i \leqslant n-1$ and $\langle 0, j \rangle = q_0$ if $j \geqslant n$. For $t \geqslant 1$ and $k \in \mathbf{Z}$ we have $\langle t, k \rangle = c_t(k)$. The idea of building B^{g} is to obtain a succession of "more and more" grouped configurations. $c_{\mathscr{C}}[u]$ the initial configuration is γ_α^n-grouped. We then proceed through a kind of backward induction requiring the next configurations to be $\gamma_\alpha^{n-q}, \gamma_\alpha^{n-2q}, \ldots, \gamma_\alpha^{n-tq}$-grouped. At time $\lfloor n/q \rfloor$, we then have a $\gamma^{n-q\lfloor n/q \rfloor}$, that is $\gamma_\alpha^{n \bmod q}$-grouped configuration. Besides that, in order to lose less time, we require B^{g} to compute in the same time it groups. Formally, we want B^{g} to yield the sequence (σ_t) for $t \in \{0, 1, \ldots, \lfloor n/q \rfloor\}$ of configurations defined as follows:

$$\sigma_t = \gamma_\alpha^{n-tq}(G_f^{(t)}(c_{\mathscr{A}}[u])), \tag{2}$$

where[7] $\sigma_0 = c_{\mathscr{C}}[u] = \gamma_\alpha^n(c_{\mathscr{A}}[u])$ is the initial configuration. With (σ_t) is associated a space–time diagram the sites of which we shall denote by $[t, k]$. For $t \in \{0, 1, \ldots, \lfloor n/q \rfloor\}$ and $k \in \mathbf{Z}$ we have $[t, k] = \sigma_t(k)$. Figure 5 shows an example of such a diagram with

[7] We assume that $G_f^{(0)} = \mathrm{Id}$.

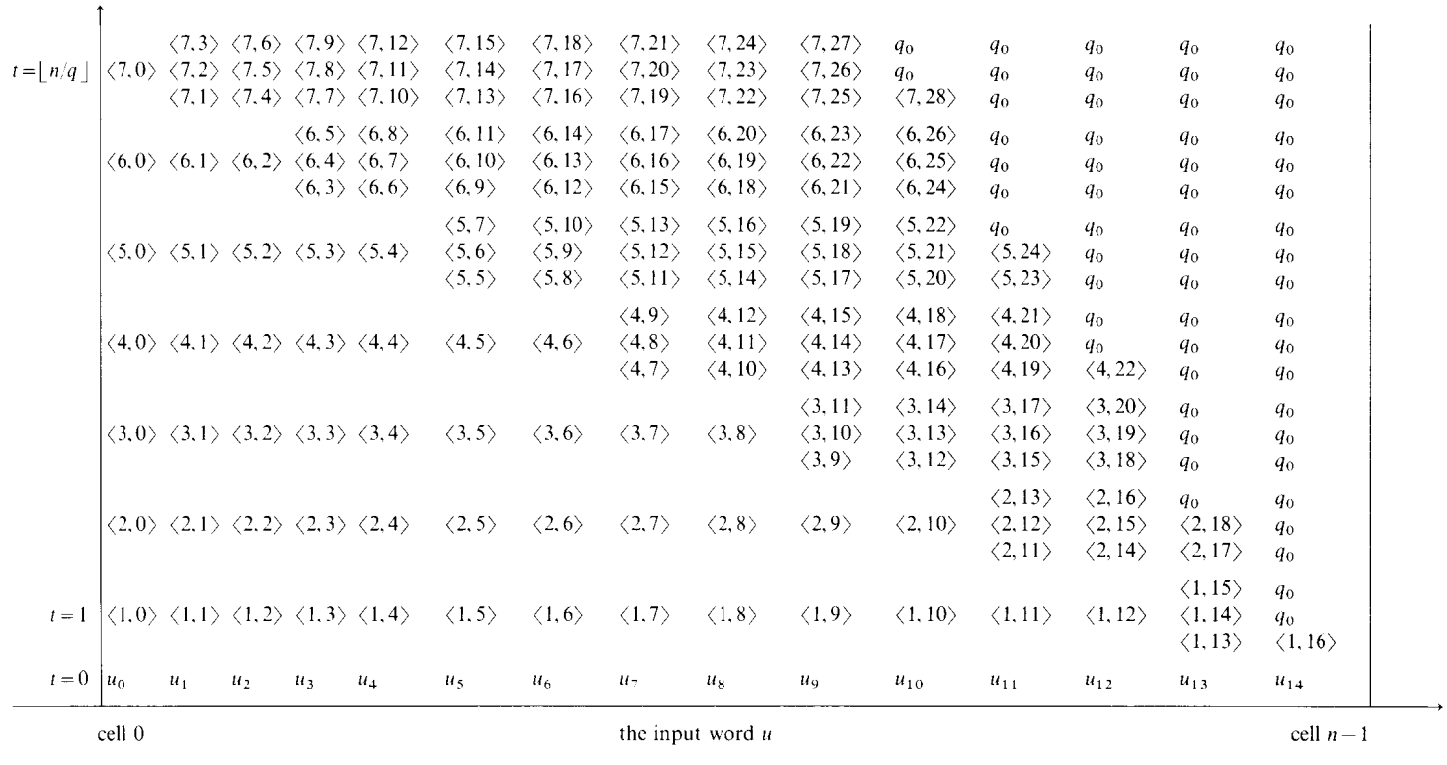

Fig. 5.

$\alpha=3$, $q=2$ and $n=15$. To complete what is said above, some remarks have to be made.

(1) Our desire to compute while grouping takes place forces B^g to behave exactly like $\mathscr{A}$ in the part of the diagram where sites are not yet "grouped" (for $0 \leqslant k < n-tq$). The transition function of B^g must, therefore, "do" exactly the same thing as $\mathscr{A}$'s one in that area and, thus, must be provided with at least as much information. This is the reason why we have to require q the range of B^g to be greater or equal to $\mathscr{A}$'s one: p.

(2) From instant 1 to $\lfloor n/q \rfloor$, we pass, at each step, from a γ_α^x-grouped configuration to a γ_α^{x-q}-grouped one. At time $\lfloor n/q \rfloor$ we obtain a $\gamma_\alpha^{n \bmod q}$-grouped configuration. If we continue the same way up to time $\lfloor n/q \rfloor + 1$, we shall obtain a $\gamma_\alpha^{n \bmod q - q}$-grouped configuration. But since $n \bmod q < q$, then $n \bmod q - q < 0$, and γ_α^k is not defined for $k<0$. This is why we have to stop at time $\lfloor n/q \rfloor$, thus only achieving a $\gamma_\alpha^{n \bmod q}$-grouping and leaving the first $n \bmod q$ cells ungrouped.

(3) The backward induction described above induces a kind of grouping "wave" which appears clearly on Fig. 5. In our construction, this wave propagates leftwards, but we could very well imagine the opposite situation in which it would start from the left end of the word u and propagate rightwards. By the way, we would be rid, in such a case, of the problem of the $n \bmod q$ ungrouped cells at time $\lfloor n/q \rfloor$. However, there is a good reason why we did not choose that option. Since we require to compute while grouping takes place, we increase the nonquiescent part of the successive configurations of $\mathscr{A}$ by p cells on the right at each time step. If the grouping wave was going in the same direction, it would never be able to catch up with the right end of the nonquiescent part of the line if q was equal to p (if we had $q>p$ then the grouping wave would be able to catch up with it but only at time $\lfloor n/(q-p) \rfloor$). And it is important to allow such an equality, since we want to be able to obtain a speeded-up automaton without any increase of the range (cf. Introduction). By having the grouping wave go leftwards we cause all the increase of the nonquiescent part to occur in the grouped part of each configuration, thus avoiding the problem and allowing $p=q$.

(4) A striking feature of our grouping process is that $\lfloor n/q \rfloor$, the grouping time, is independent of p and α. Remarks (1) and (3) give some insight into the reason why it is independent of p. However, giving a synthetic and intuitive reason for its α-independence is much harder. The proof of the existence of B^g which we give below contains all the elements needed to explain it but they are hidden and scattered among the calculations. The best intuitive approach we can provide is to look at the example of Fig. 5 and see that the size of the groups of states we construct is somehow orthogonal to the core of the grouping process (we hint at this by representing these groups as vertical *piles*). If we change the value of α, the overall shape of the figure will not change significantly; simply the configurations will become a bit more or a bit less thick but the succession of leaps of q cells done by the grouping "wave" will not be altered and this is what determines the grouping time $\lfloor n/q \rfloor$.

To prove the existence of B^g we might have listed in detail its alphabet and transition function and then proved (by induction on t for instance) that the global map thus induced fulfilled the requirements embodied by the sequence (σ_t). However,

we feel that such an approach does not provide a good understanding of the key aspects of the construction. We shall rather proceed in the opposite way. The sequence (σ_t) may be seen as the specification of a global map. Of course, it is not a complete specification but any global map fulfilling it would do. If we prove that such a global map may be computed by fully local means, that is exactly as a local map would do, but without necessarily giving such a function explicitly, the existence of B^g will be proven. We shall base our argument on space–time-diagram considerations, showing that the state of any site $[t,k]$ may be computed from only the data contained in the sites $[t-1,k-q]$ to $[t-1,k+q]$ (B^g is of range q) and possibly from constants. As defined in equation (2) the sequence (σ_t) yields an STD with the following structure: for any t such that $0\leqslant t\leqslant \lfloor n/q \rfloor$ and any $k\in\mathbf{Z}$ we have

$$[t,k]=\begin{cases}(\lambda,\ldots,\lambda)_\alpha & \text{if } k<0,\\ \langle t,k\rangle & \text{if } 0\leqslant k<n-tq,\\ (\langle t,I\rangle,\langle t,I+1\rangle,\ldots,\langle t,I+\alpha-1\rangle), & \\ \quad \text{with } I=n-tq+(k-(n-tq))\alpha & \text{if } k\geqslant n-tq.\end{cases}$$

In fact, for $k\geqslant 0$ (we do not care about negative cells since it is clear that we can arrange B^g so as to put them all in state $(\lambda,\ldots,\lambda)_\alpha$), the state of each site $[t,k]$ is a sequence of site $\langle t,i\rangle$ for i such that $Min(t,k)\leqslant i\leqslant Max(t,k)$ with Min and Max defined as follows:

$$\begin{aligned} Min(t,k)&=\begin{cases}k & \text{if } 0\leqslant k<n-tq,\\ n-tq+(k-(n-tq))\alpha & \text{if } k\geqslant n-tq,\end{cases}\\ Max(t,k)&=\begin{cases}k & \text{if } 0\leqslant k<n-tq,\\ n-tq+(k-(n-tq))\alpha+\alpha-1 & \text{if } k\geqslant n-tq.\end{cases}\end{aligned} \tag{3}$$

To be able to compute $[t,k]$, that is the sequence of sites $\langle t,i\rangle$ for all i such that $Max(t,k)\leqslant i\leqslant Min(t,k)$, the following three conditions must be fulfilled:

(1) We have to be able to decide if $[t,k]$ is to contain one or α sites of $\mathscr{A}$, i.e. whether we are in the case $0\leqslant k<n-tq$ or $k\geqslant n-tq$.

(2) We have to be provided with all the necessary information to compute each of the $\langle t,i\rangle$ through f, i.e. all sites $\langle t-1,i+j\rangle$ for all j such that $-p\leqslant j\leqslant p$. The union of the sets of sites of $\mathscr{A}$ contained in the sites $[t-1,k-q]$ to $[t-1,k+q]$ must, therefore, contain the set

$$\begin{aligned}&\{\langle t-1,\ i+j\rangle,\ Min(t,k)\leqslant i\leqslant Max(t,k),\ -p\leqslant j\leqslant p\}\\ &\quad=\{\langle t-1,i\rangle,\ Min(t,k)-p\leqslant i\leqslant Max(t,k)+p\}.\end{aligned}$$

This union is simply the set

$$\{\langle t,i\rangle, Min(t-1,k-q)\leqslant i\leqslant Max(t-1,k+q)\};$$

our condition thus boils down to:

$$Min(t-1,k-q) \leqslant Min(t,k)-p,$$

$$Max(t,k)+p \leqslant Max(t-1,k+q).$$

In many cases (see the example of Fig. 5) the above inequalities are *strict* inequalities. In other words, the sites $[t-1,k-q]$ to $[t-1,k+q]$ contain *more* sites of $\mathscr{A}$ than needed in order to compute the content of $[t,k]$; therefore,

(3) We have to be able to choose the relevant sites of $\mathscr{A}$ among those contained in the sites $[t-1,k-q]$ to $[t-1,k+q]$. By chance (!!), we have the following relation:

$$Max(t-1,k+q)-(Max(t,k)+p)=q-p,$$

whatever $k \geqslant 0$ and $t \geqslant 1$. If condition (2) is fulfilled then we will know that the sequence of sites we "need" is contained in the sequence we "have". This last result tells us that the end of the sequence we "need" is situated $q-p$ positions before the end of the sequence we "have". Since we know the length of the sequence we "need", ($2p+1$ or $2p+\alpha$ depending on the answer to the question of condition (1)) we may, therefore, localize and extract its elements from those of the sequence we "have".

The solution of condition one is the following: If all sites $[t-1,k-q]$ to $[t-1,k+q]$ contain only one site of $\mathscr{A}$ each then $[t,k]$ contains also one site. If at least one of the sites $[t-1,k-q]$ to $[t-1,k+q]$ contains α elements of $\mathscr{A}$ then $[t,k]$ contain also α sites of $\mathscr{A}$. In effect, if all sites $[t-1,k-q]$ to $[t-1,k+q]$ contain only one site each that means that they are in the ungrouped part of the diagram and specifically $[t-1,k+q]$ is. That means that $k+q<n-q(t-1)$; so, $k<n-tq$, and $[t,k]$ is also in the ungrouped part of the diagram and, therefore, contains only one site. The second part of the above statement is obtained the same way. The solution of conditions (2) and (3) rely on the following proposition.

Proposition 5.1. *Let Min and Max be two functions defined as in equation* (3) *and p an integer such that* $p \leqslant q$ *then for all t such that* $1 \leqslant t \leqslant \lfloor n/q \rfloor$ *and all* $k \geqslant 0$ *we have*

$$\begin{aligned} &Min(t-1,k-q) \leqslant Min(t,k)-p, \\ &Max(t,k)+p \leqslant Max(t-1,k+q), \\ &Max(t-1,k+q)-(Max(t,k)+p)=q-p. \end{aligned} \tag{4}$$

Proof. (1) If $0 \leqslant k < n-tq$ then $Min(t,k)=k$ and $Max(t,k)=k$. We have $k-q<n-(t-1)q$ and $k+q<n-(t-1)q$; so, $Min(t-1,k-q)=k-q$ and $Max(t-1,k+q)=k+q$. Equations (4) are obviously fulfilled.

(2) If $n-tq \leqslant k < n-tq+2q$, we have $n-tq \leqslant k$; so, $Min(t,k)=I$ and $Max(t,k)=I+\alpha-1$, with $I=n-tq+(k-(n-tq))\alpha$. $k \geqslant n-tq$ entails $k+q \geqslant n-(t-1)q$; so,

$$\begin{aligned} Max(t-1,k+q) &= n-(t-1)q+(k+q-(n-(t-1)q))\alpha+\alpha-1 \\ &= n-tq+q+(k-(n-tq))\alpha+\alpha-1 \\ &= I+q+\alpha-1. \end{aligned}$$

We then have

$$Max(t,k)+p \leqslant Max(t-1,k+q),$$

$$Max(t-1,k+q)-(Max(t,k)+p)=q-p.$$

Since, on the other hand, $k<n-tq+2q$, we have $k-q<n-(t-1)q$; so, $Min(t-1,k-q)=k-q$. So, $Min(t,k)-p-Min(t-1,k-q)=n-tq+(k-(n-tq))\alpha-p-k+q$. Since $\alpha \geqslant 1$ and $q \geqslant p$, this is positive; so,

$$Min(t-1,k-q) \leqslant Min(t,k)-p.$$

(3) If $k \geqslant n-tq+2q$ then we have $k \geqslant n-tq$; so, the computations for the function Max are the same as above. However, we have $k-q \geqslant n-(t-1)q$; so, $Min(t-1, k-q)=n-(t-1)q+(k-q-(n-(t-1)q))\alpha=n-tq+(k-(n-tq))\alpha+q+2q\alpha=I+q(1-2\alpha)$. Since $\alpha \geqslant 1$, $q \geqslant p$ and $Min(t,k)=I$, we have

$$Min(t-1,k-q) \leqslant Min(t,k)-p. \qquad \square$$

We have thus shown that there exists a B^g fulfilling our requirements. In fact, we have simply proved the following proposition.

Proposition 5.2. *There is a cellular automaton of range $q \geqslant p$ whose global map turns in one step any configuration $\gamma_\alpha^x(c)$, with $x \geqslant q$ and c a configuration of $\mathscr{A}$, into the configuration $\gamma_\alpha^{x-q}(G_f(c))$.*

The proof of this proposition is exactly the same as that of the existence of B^g since, in that case, the fact that $x=n-tq$ is irrelevant. We may generalize this result in the following way.

Proposition 5.3. *For any z such that $-q \leqslant z \leqslant q$ there is an automaton of range $q \geqslant p$ whose global map turns in one step any configuration $\gamma_\alpha^x(c)$, for any $x \geqslant z$ and c a configuration of $\mathscr{A}$, into the configuration $\gamma_\alpha^{x-z}(G_f(c))$, provided that at least all the active cells of $\gamma_\alpha^{x-z}(G_f(c))$ (those which are not in state $(\lambda,\ldots,\lambda)_\alpha$ or $(q_0,\ldots,q_0)_\alpha$) can know locally that they are the active cells and can compute the value of z.*

To prove the existence of B^g (alias Proposition 5.2) we showed that, to pass from $\gamma_\alpha^x(c)$ to $\gamma_\alpha^{x-q}(G_f(c))$ the kind of *shift-and-compute* (shift: $\gamma_\alpha^x \to \gamma_\alpha^{x-q}$, compute: $c \to G_f(c)$) procedure that has to be done may be done locally, i.e. such that each cell is provided with all the necessary information to perform it. In that case, the index of the shift (the z of Proposition 5.3) is a constant (q) and may, therefore, be provided to *all* cells by embedding it implicitly in the transition function. On the contrary, if we want z not to be a constant, i.e. if we want it to depend on informations contained in c or in a configuration prior to c (we shall see an application of that below), then we will almost certainly be unable to provide all cells with the necessary information to compute z. However, since we are working with configurations of $\mathscr{A}$, we know that

almost all cells of $\gamma_\alpha^{x-z}(G_f(c))$ are in state $(\lambda,\ldots,\lambda)_\alpha$ or $(q_0,\ldots,q_0)_\alpha$ except only a finite number lying between cell 0 and a certain cell y. Only the cells belonging to this "active part" really need to do the shift-and-compute work, while the others only have to know that they have nothing to do. By doing the following:

- provide at least all active cells with the message "you are active",
- provide all cells which have been told they were active with the necessary information to compute z,

we put all cells which are told they are active in the same situation as all cells in the case of the proof of Proposition 5.2. We can then prove that these cells may accomplish the shift-and-compute work in these conditions exactly in the same way as in the proof of the existence of B^g with the three conditions and an equivalent of Proposition 5.1 for the calculations. If more cells than those which are really active receive the message "you are active", it does no harm since for these cells the shift-and-compute procedure will simply result in the production of a $(q_0,\ldots,q_0)_\alpha$. On the other hand, all cells which do not receive this message will, therefore, know that they just have to be in state $(\lambda,\ldots,\lambda)_\alpha$ or $(q_0,\ldots,q_0)_\alpha$ (they can decide which one from the context).

We shall use the result of the Proposition 5.3 in order to obtain the last configuration of B^G. At time $\lfloor n/q \rfloor$, B^g yields the configuration

$$\sigma_{\lfloor n/q \rfloor} = \gamma_\alpha^{n \bmod q}(G_f^{(\lfloor n/q \rfloor)}(c_{\mathscr{A}}[u])),$$

and, at the next instant, we want B^G to yield the configuration

$$\sigma_{\lfloor n/q \rfloor + 1} = \gamma_\alpha^{0}(G_f^{(\lfloor n/q \rfloor + 1)}(c_{\mathscr{A}}[u])).$$

Since $\gamma_\alpha^0 = \gamma_\alpha$, we shall have achieved our goal and obtained a fully grouped configuration. Proposition 5.3 tells us that we can pass from $\sigma_{\lfloor n/q \rfloor}$ to $\sigma_{\lfloor n/q \rfloor + 1}$ with a $z = n \bmod q$ (we have $-q \leqslant n \bmod q \leqslant q$) but only if at least all active cells of $\sigma_{\lfloor n/q \rfloor + 1}$ are provided with the value of $n \bmod q$ and told they are active in some way. The configuration $G_f^{(\lfloor n/q \rfloor + 1)}(c_{\mathscr{A}}[u])$ has at most $n + p(\lfloor n/q \rfloor + 1)$ nonquiescent cells. Since $p \leqslant q$, we have $n + p(\lfloor n/q \rfloor + 1) \leqslant n + q(\lfloor n/q \rfloor + 1) = 2n - n \bmod q + q$. The number of cells of $\sigma_{\lfloor n/q \rfloor + 1}$ which are not in state $(\lambda,\ldots,\lambda)_\alpha$ or $(q_0,\ldots,q_0)_\alpha$ (the active cells of $\sigma_{\lfloor n/q \rfloor + 1}$), is, therefore, at most

$$\left\lceil \frac{2n - n \bmod q + q}{\alpha} \right\rceil.$$

Since $\alpha \geqslant 2$ we have

$$\left\lceil \frac{2n - n \bmod q + q}{\alpha} \right\rceil \leqslant \left\lceil n + \frac{q - n \bmod q}{\alpha} \right\rceil \leqslant n + q - n \bmod q.$$

We have, therefore, to provide all cells 0 to $n + q - n \bmod q - 1$ at time $\lfloor n/q \rfloor + 1$ with

- the value of $n \bmod q$, and
- a message that will tell them that they are active.

The first of these two conditions will be treated in Section 6. The second, which amounts to synchronizing all cells 0 to $n+q-n \bmod q-1$ at time $\lfloor n/q \rfloor + 1$, will be treated in Section 7. In Section 8 we shall gather together all the components worked out in Sections 5–7 to build B^G and then compose it with the B^* of Section 4 (the weak form) in order to obtain the strong form of the speed-up theorem. This composition operation will also rely on the synchronization process whose role is, therefore, twofold.

6. Computing $n \bmod q$

We said, at the end of the preceding section, that, in order to compute the last configuration of B^G, we needed to provide all its active cells, i.e. all cells 0 to $n+q-n \bmod q-1$, with the value of $n \bmod q$ at time $\lfloor n/q \rfloor + 1$. We shall show here how to do that.

Proposition 6.1. *There exists a cellular automaton* $B^m=(Q^m, q, g^m)$ *with* $U \cup \{(\lambda,\ldots,\lambda)_\alpha, (q_0,\ldots,q_0)_\alpha\} \subset Q^m$ *and* $\{1,2,\ldots,q-1\} \subset Q^m$ *(we assume that* $\{1,2,\ldots,q-1\} \cap U = \emptyset$*) which, when fed with the initial configuration* $c_{\mathcal{G}}[u]$ *for a word of length* $n \geqslant q$*, yields after* $\lfloor n/q \rfloor$ *time steps a configuration in which cells* $n \bmod q$, $n \bmod q+1,\ldots, n-n \bmod q-1$ *are in state* $n \bmod q$.

Remark. The state at time $\lfloor n/q \rfloor + 1$ of each of the cells 0 to $n+q-n \bmod q-1$ depends on the state of at least one of cells $n \bmod q$ to $n-n \bmod q-1$ at time $\lfloor n/q \rfloor$ which is $n \bmod q$. In that sense we may say that cells 0 to $n+q-n \bmod q-1$ are provided with the value of $n \bmod q$ at time $\lfloor n/q \rfloor + 1$. On the other hand, we limit our scope here to words of length $n \geqslant q$ since, if $n<q$, then we have $\lfloor n/q \rfloor = 0$; B^m does not have enough time to perform even one step. We shall treat the case $n<q$ in Section 8 when considering the final solution to the grouping problem.

Proof. We shall only give a brief sketch of the proof of the above result and leave the details to the reader since it is straightforward. Let us denote $[t,k]^m$ the sites of the diagram associated with the computation of B^m on the initial configuration $c_{\mathcal{G}}[u]$. Figure 6 shows a "continuous approximation" of the kind of diagram we wish to obtain. We have two signals G and D starting from both ends of the word u and heading towards the middle at the fastest possible speed. They are composed of sites $[t, tq-1]^m$ and $[t, n-tq]^m$, respectively, for $t \in \{0,1,2,\ldots,\lfloor n/2q \rfloor\}$, which may be marked with a special state for instance. The distance between the points of G and D at time $\lfloor n/2q \rfloor$ is

$$n-q\lfloor n/2q \rfloor - (q\lfloor n/2q \rfloor - 1) = n-(n-n \bmod 2q)+1$$

$$= n \bmod 2q+1.$$

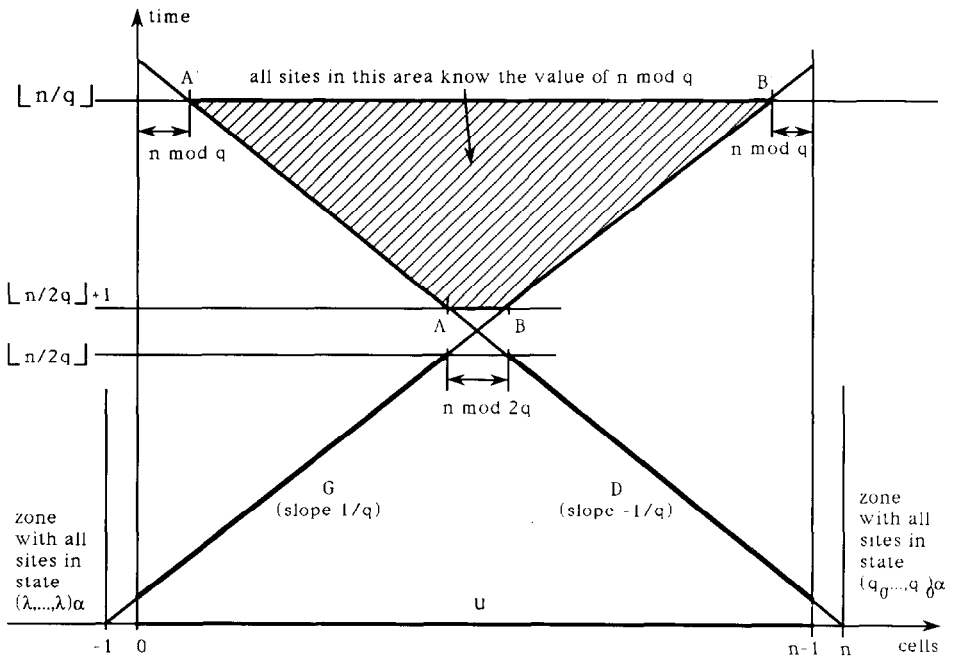

Fig. 6.

This distance is less than or equal to $2q$ but still positive at time $\lfloor n/2q \rfloor$. Since G and D proceed by leaps of q cells at each time step this implies that the lines which are the extensions of the two signal cross between instant $\lfloor n/2q \rfloor$ and the next. The points

$$A = [\lfloor n/2q \rfloor + 1, n - q(\lfloor n/2q \rfloor + 1)]^m$$

and

$$B = [\lfloor n/2q \rfloor + 1, q(\lfloor n/2q \rfloor + 1) - 1]^m$$

prolong D and G, respectively, at time $\lfloor n/2q \rfloor + 1$. All sites between A and B (included) see the two extremities of G and D at time $\lfloor n/2q \rfloor$ and may, therefore, evaluate the distance between them. They obtain this way the value of $n \bmod 2q$, from which they can deduce that of $n \bmod q$ (if $n \bmod 2q \geqslant q$ then $n \bmod q = n \bmod 2q - q$, else $n \bmod q = n \bmod 2q$). We can, therefore, arrange g^m so as to put them in state $n \bmod q$. This state then diffuses to the largest possible number of sites, i.e. to those inside the trapezoid marked by the extension of signal D and G (these two lines crossed between instant $\lfloor n/2q \rfloor$ and $\lfloor n/2q \rfloor + 1$). At time $\lfloor n/q \rfloor$ all cells between site

$$A' = [\lfloor n/q \rfloor, n - q\lfloor n/q \rfloor]^m = [\lfloor n/q \rfloor, n \bmod q]^m$$

and

$$B' = [\lfloor n/q \rfloor, q\lfloor n/q \rfloor - 1]^m = [\lfloor n/q \rfloor, n - n \bmod q - 1]^m$$

(included) are in state $n \bmod q$. The desired result is, therefore, obtained.

Note: In the degenerate case $q \leqslant n < 2q$ we have $\lfloor n/2q \rfloor = 0$. The points of G and D at that time are the two extremities of u that is to say the last λ before the first letter of u and the first q_0 after the last letter of u, respectively. We also have $\lfloor n/2q \rfloor + 1 = \lfloor n/q \rfloor = 1$. Points A and A' are the same and so are B and B'. □

7. Synchronization

We shall need synchronization for two purposes. First, to finish grouping and to enable B^G to yield a completely (i.e. γ_x^0) grouped configuration, we saw that at least all the active cells[8] of its last configuration must be synchronized at time $\lfloor n/q \rfloor + 1$ (see Section 5). Second, to compose B^G, the grouping automaton, and B^*, the speeded-up automaton, i.e. to build an automaton which will work like B^G until a certain time and then like B^*, we must be able to tell cells to "change mode of operating" at that time. But we shall see in Section 8, when finally building B^G, that only the active cells of the common configuration (the last of B^G and the first of B^*) need to be given that order. For these two purposes, it is therefore sufficient to have cells 0 to $n+q-n \bmod q-1$, the active cells of the last configuration of B^G, synchronized at time $\lfloor n/q \rfloor + 1$.

We shall achieve this goal in the following way: We shall prove that there exists an automaton $B^s = (Q^s, q, g^s)$ (we take B^s of range q, i.e. of the same range as B^g) such that, when starting from the initial configuration $c_{/}[u]$, a certain group of cells are at time $\lfloor n/q \rfloor$ in a special synchronizing state never taken before by any cell. These cells are spread along the line so that at least one of them is situated within range q to any of the cells 0 to $n+q-n \bmod q-1$. This way, when determining the state at time $\lfloor n/q \rfloor + 1$ of any of our cells 0 to $n+q-n \bmod q-1$, the transition function is provided with something it was never provided before (we shall call it the *synchronizing information*) and thus "knows" that we are passing from instant $\lfloor n/q \rfloor$ to instant $\lfloor n/q \rfloor + 1$. This way, our initial synchronization problem is reduced to synchronizing this special group of cells at time $\lfloor n/q \rfloor$ in the classical sense of "having them all enter at time $\lfloor n/q \rfloor$ a special state that they never took before".

7.1. The firing-squad synchronization problem (FSSP)

This classical sense is that of the FSSP (see e.g. [9, 7]) of which we use the solutions to build B^s. Let us review it briefly. We are given a finite segment of τ cells ($\tau \geqslant 2$) whose behaviour is governed by a local transition function of range 1: φ. The two end cells (since $\tau \geqslant 2$, we have two distinct end cells) are particular cells since the left one has no left neighbour and the right one has no right neighbour. Their behaviour is, therefore, given by two special (2-ary) transition functions: φ_l and φ_r. At time 1 the left end cell is set by an intervention from the "user" in a special state G (called the General state by analogy with a General commanding to a line of soldiers to shoot) whereas all other

[8] Those which are not in state $(\lambda, \ldots, \lambda)_x$ or $(q_0, \ldots, q_0)_x$.

cells, including the rightmost one, are in the quiescent state R. A sequence of finite configurations is then computed by the global map induced by φ, φ_l and φ_r. The key aspect of the problem is that we require all cells to enter together and for the first time in another special state F (for fire) at a certain instant t_s. It has been shown [9] that t_s cannot be less than $2\tau - 1$ for a segment of length τ and that there exist solutions working in minimum time [2, 7, 14]. We assume that we have such a solution, i.e. a 4-tuple $\mathcal{F} = (S, \varphi, \varphi_l, \varphi_r)$, with $S = \{s_0, s_1, \ldots, s_m\}$, $s_0 = R$, $s_1 = G$, $s_2 = F$, $\varphi : S^3 \to S$, $\varphi_l : S^2 \to S$, $\varphi_r : S^2 \to S$ such that $\varphi(R, R, R) = \varphi_l(R, R) = \varphi_r(R, R) = R$ (R is a quiescent state) and such that starting with the configuration

$$\underbrace{GRR \ldots R}_{\tau},$$

with $\tau \geqslant 2$, at time 1, we obtain, through the application of $\varphi, \varphi_l, \varphi_r$, at time $2\tau - 1$, the configuration

$$\underbrace{FF \ldots FF}_{\tau}.$$

Figure 7 shows an example of the kind of space–time diagram associated with such a process, the Y_S represent any state other than G, R or F, the X_S represent any state other than F or R. All known solutions to the FSSP which work in minimum time have, as shown in Fig. 7, a *first wave* represented here by the sequence of Y, which starts from the General cell and reaches the right-cell soldier within the smallest possible time. It is this feature which puts in the space–time diagram of the solutions to the FSSP this characteristic figure (the trapezoid $IJKL$) shaped as a Guillotine bladc.

To generalize the range 1 FSSP to the range q (q arbitrary) context of B^s we shall give a privileged role to cells $q-1, 2q-1, \ldots, q\lfloor n/q \rfloor - 1$. Since they are placed at

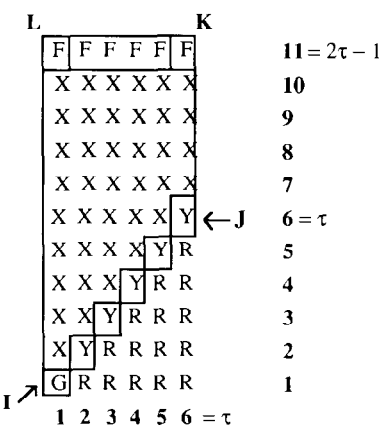

Fig. 7.

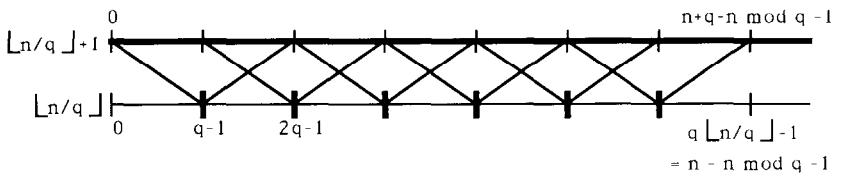

Fig. 8.

a distance q from one another, the "mutual-visibility conditions" of these cells in the context of an automaton of range q are exactly the same as those of a sequence of consecutive cells in an automaton of range 1: each cell "sees" one and only one neighbour on each side. We can, therefore, define the transition function of B^s so as to simulate the execution of an automaton of range 1 on these cells while the others (those lying between them) remain idle. Moreover, if we are able to synchronize all these cells at time $\lfloor n/q \rfloor$ (in the sense of having them all enter state F at that time and for the first time). Then we shall have achieved our goal. Since $q\lfloor n/q \rfloor - 1 = n - n \bmod q - 1$ it is clear (see Fig. 8) that the state of all cells 0 to $n+q-n \bmod q-1$ at time $\lfloor n/q \rfloor + 1$ depend on the state of at least one cell of the sequence $q-1, 2q-1, \ldots, q\lfloor n/q \rfloor - 1$ and that, therefore, all cells 0 to $n+q-n \bmod q-1$ are "irrigated" by the synchronizing information at time $\lfloor n/q \rfloor + 1$.

We want, therefore, to simulate on cells $q-1, 2q-1, \ldots, q\lfloor n/q \rfloor - 1$ the synchronization process of a segment of $\tau = \lfloor n/q \rfloor$ cells. The initiation of the synchronization process takes place at instant 1 of B^s which, therefore, corresponds to the instant 1 of the simulated process. Since we want the synchronization to occur at time $\lfloor n/q \rfloor$ it means that we want to synchronize in time τ a segment of length τ, which seems impossible with the solutions of the classical FSSP described above (which work at best in time $2\tau - 1$). However, in the classical FSSP, the initiation of the synchronization process takes place at only one end of the segment, whereas we shall see that in our case we can start it from both ends. We shall see below that we can put *both* end cells of our simulated segment in state G since they each "see" at that time one end of the word u and that tells them that they are at time 1. To build B^s we shall, therefore, use the variant of the FSSP in which both ends of the segment are in state G at time 1 (we shall call it the 2E-FSSP for *two-ends*-FSSP) and for which there exist solutions which synchronize a segment of length τ in time τ.

7.2. *The 2E-FSSP*

The specifications of our problem now becomes: At time 1 we have a segment of length $\tau \geqslant 2$ (it is necessary to have $\tau \geqslant 2$ to have two separate end cells) in the configuration

$$\underbrace{GRR \ldots RG}_{\tau}.$$

By applying to it the global map induced by three local functions of range 1: ψ, ψ_l and ψ_r we want to obtain at a certain time $t_s(\tau)$ the configuration

$$\underbrace{FFF\ldots FF}_{\tau}$$

so that, at any time before $t_s(\tau)$, no cell was ever in state F. Minsky [9] showed that the synchronization time of a solution to the classical FSSP may not be under $2\tau - 1$ for a segment of length τ. We show here that, in the case of the 2E-FSSP, this lower bound is τ for a segment of length τ.

Theorem 7.1. *All solutions to the 2E-FSSP synchronize in time $t_s(\tau) \geqslant \tau$ for all $\tau \geqslant 2$.*

Proof. The proof we give here is very similar to that of Minsky. Let us suppose that we have a solution to the 2E-FSSP such that $t_s(\tau_0) < \tau_0$ for some $\tau_0 \geqslant 2$. We know that at time $t_s(\tau_0)$ cell 1 is in state F. The state of cell 1 at time $t_s(\tau_0)$ depends on the states of cells 1 to $t_s(\tau_0)$ at time 1 and only on them (see Fig. 9). Since $t_s(\tau_0) < \tau_0$, the states of these cells are G for cell 1 and R for all others. So, if we start from a segment of length $\tau > \tau_0$, the state of cells 1 to $t_s(\tau_0)$ at time 1 will not be changed. Cell 1 is, therefore, also in state F at time $t_s(\tau_0)$ in the case of the synchronization of a segment of length $\tau > \tau_0$. This means that we have $t_s(\tau) \leqslant t_s(\tau_0)$ for all $\tau > \tau_0$ since if we had $t_s(\tau) > t_s(\tau_0)$ for some $\tau > \tau_0$ then we would have a cell in state F before synchronization time (cell 1 at time $t_s(\tau_0)$) and, hence, this would not be a proper synchronization. If we now take $\tau = 2\tau_0 + 1$ then we know that cell $\tau + 1$ is in state R at any time between 1 and τ_0 since R is a quiescent state (see Fig. 10). But, since $2\tau_0 + 1 > \tau_0$, we have $t_s(\tau) < t_s(\tau_0) < \tau_0$. So, cell $\tau_0 + 1$ must be in state F at some time $t_s(\tau) < \tau_0$. A contradiction. So, there is no solution to the 2E-FSSP such that there is a $\tau_0 \geqslant 2$ such that $t_s(\tau_0) < \tau_0$. □

We shall now show that there is a solution in minimal time to the 2E-FSSP. The basic idea is to cut the line in two halves of length approximately $\tau/2$ and synchronize them with two copies of a solution to the classical FSSP propagating symmetrically

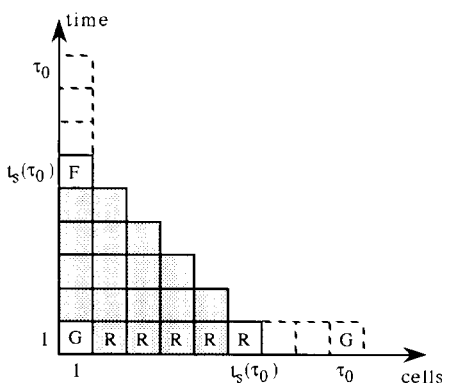

Fig. 9.

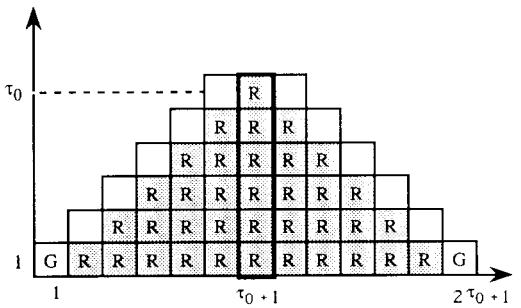

Fig. 10.

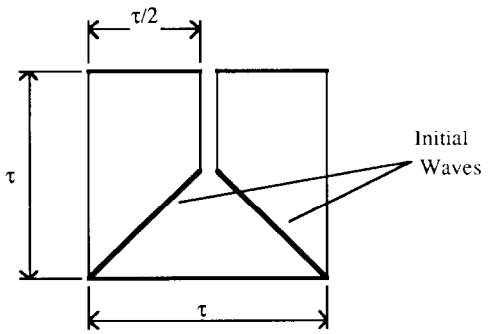

Fig. 11.

from the G at the two ends of the segment (see Fig. 11). The synchronization time of the total segment is, therefore, reduced to the time taken by a solution to the classical FSSP to synchronize a half segment, i.e. approximately $2(\tau/2)=\tau$. Our goal is to obtain a solution $\mathcal{G}=(T,\psi,\psi_l,\psi_r)$ with $T=\{z_0,z_1,\ldots,z_x\}$, $z_0=R$, $z_1=G$, $z_2=F$, $\psi: T^3\to T$, $\psi_l: T^2\to T$, $\psi_r: T^2\to T$, such that $\psi(R,R,R)=\psi_l(R,R)=\psi_r(R,R)=R$ (R is a quiescent state) and which fulfills the requirements of the 2E-FSSP (see the beginning of this subsection). We shall obtain $\mathcal{G}$ from $\mathcal{F}$, our solution to the classical FSSP, through a kind of approximation process yielding a sequence $\mathcal{G}^1,\mathcal{G}^2,\ldots,\mathcal{G}^5$ of solutions which will come gradually closer to what we want and whose last element will be $\mathcal{G}$. To each $\mathcal{G}^i$ is associated an alphabet T^i and three transition functions ψ^i, ψ^i_l and ψ^i_r.

We shall first concentrate on the left half of the segment and provide a solution only adequate for the case where the length of the segment τ is an odd number. However, we shall take care that this solution does not put any cell into state F before time τ in the case τ even. This way, we shall be able to complete it later in order to obtain synchronization in time τ also in that case.

Proposition 7.2. *There exists a* $\mathcal{G}^1$ *which*

(1) *synchronizes cells* 1 *to* $\lfloor \tau/2 \rfloor + 1$ *at time* τ *if* τ *is odd and* $\tau \geqslant 3$ (*the smallest value for* τ *odd and* $\tau \geqslant 2$);

(2) *does not put any cell in state* F *before time* τ *if* τ *is even and* $\tau \geqslant 2$.

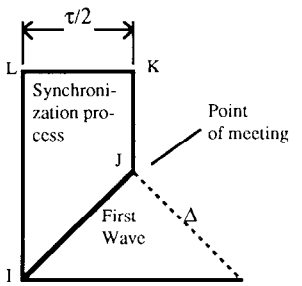

Fig. 12.

Proof. We build $\mathscr{G}^1$ from $\mathscr{F}$ in the following way (see Fig. 12). From the G in the left-end cell starts a synchronization process identical to that of $\mathscr{F}$. From the G in the right-end cell comes a signal Δ which propagates leftwards as fast as possible (of one cell to the left at each time step). When the first wave of the synchronization process meets with Δ, at approximately the middle of the segment, the cell belonging to the first wave at that instant and involved in the meeting is marked and then treated as if it was the right end of the segment at the next instants. This way, we cause the synchronization process to be contained in the left part of the segment. It is already clear, at that point, that we will manage to synchronize this way at time τ only if τ is odd since the length of the segment $[IL]$ (which is equal to $2[LK]-1$ since the synchronization process we use is a minimal-time solution to the classical FSSP) is necessarily an odd number. Let us prove that we indeed obtain that way, the announced behaviour.

At time $t \geqslant 1$, the first wave of the synchronization process has reached cell t. At time 1, the signal Δ is materialized by the G in the right-end cell (cell number τ). At any of the following instants $t \geqslant 2$, it may be embodied by the presence in cell $\tau - t + 1$ of a special state $\oplus$ added to the alphabet of $\mathscr{G}^1$ for that purpose. At time $\lfloor \tau/2 \rfloor + 1$ the first wave is in cell $\lfloor \tau/2 \rfloor + 1$. At the previous instant, Δ was in cell $\tau - \lfloor \tau/2 \rfloor + 1$. We have $(\tau - \lfloor \tau/2 \rfloor + 1) - (\lfloor \tau/2 \rfloor + 1) = \tau - 2\lfloor \tau/2 \rfloor = r \bmod 2 = 0$ or 1. So, whatever the parity of τ, the cell of the first wave at time $\lfloor \tau/2 \rfloor + 1$ (cell $\lfloor \tau/2 \rfloor + 1$) "sees" the cell of Δ at time $\lfloor \tau/2 \rfloor$ (cell $\tau - \lfloor \tau/2 \rfloor + 1$). The "marking of the right end" operation, alluded to above, therefore takes place in cell $\lfloor \tau/2 \rfloor + 1$ at time $\lfloor \tau/2 \rfloor + 1$. It is conducted in the following way: To T^1, the alphabet of $\mathscr{G}^1$, we add a subset $\bar{S} = \{\bar{s}, s \in S\}$ disjoint of S, the alphabet of $\mathscr{F}$, but in bijection with it. The overbar on the state of a cell will mean that this cell plays the role of the right-end cell. At time $\lfloor \tau/2 \rfloor$ (the instant preceding the marking) cell $\lfloor \tau/2 \rfloor$ is in a certain state $y \in S$. According to the above "visibility" result, we know that one of the cells $\lfloor \tau/2 \rfloor + 1$ or $\lfloor \tau/2 \rfloor + 2$ contains a state materializing the presence of Δ ($\oplus$ or G if $\lfloor \tau/2 \rfloor = 1$). To reflect the fact that cell $\lfloor \tau/2 \rfloor + 1$ is now the right-end cell, we use φ_r to compute its state and we add an overbar to it so as to mark it as the right-end cell. We arrange the transition function of G^1 so as to put cell $\lfloor \tau/2 \rfloor + 1$ at time $\lfloor \tau/2 \rfloor + 1$ in state $\overline{\varphi_r(y, r)}$. All the possible situations appear in Fig. 13. This way, we have at time $\lfloor \tau/2 \rfloor$ on cells 1 to $\lfloor \tau/2 \rfloor + 1$ a segment of length

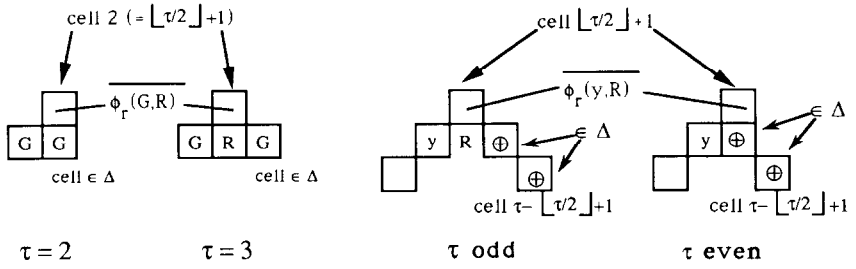

Fig. 13.

$\lfloor \tau/2 \rfloor + 1$ in the state in which it would be (if we except the overbar on the state of the right-end cell) after $\lfloor \tau/2 \rfloor + 1$ steps of the execution of $\mathscr{F}$. We continue the execution of the synchronization process on this reduced segment by making the transition function of $\mathscr{G}^1$ treat the cell which is in a marked state as if it was the right-end cell. For any $x, y, z \in S$, we let

$$\psi^1(x, y, \bar{z}) = \varphi(x, y, z),$$

$$\psi_l^1(x, \bar{y}) = \varphi_l(x, y); \text{ this is useful if } \tau = 2 \text{ or } 3,$$

$$\psi^1(x, \bar{y}, z) = \overline{\varphi_r(x, y)},$$

$$\psi_r^1(x, \bar{y}) = \overline{\varphi_r(x, y)}; \text{ this is useful if } \tau = 2.$$

Besides that, we ensure that all cells beyond cell $\lfloor \tau/2 \rfloor + 1$ are in state R and we make an exception to the above rules for the state F: if ever $\bar{F}$ had to appear according to them, then just F appears instead. In these conditions, our segment of $\lfloor \tau/2 \rfloor + 1$ cells will synchronize (all its cells will enter state F for the first time) at time $2(\lfloor \tau/2 \rfloor + 1) - 1 = 2\lfloor \tau/2 \rfloor + 1$. If τ is odd then we have $\tau = 2\lfloor \tau/2 \rfloor + 1$; we have definitely achieved in that case the synchronization of cells 1 to $\lfloor \tau/2 \rfloor + 1$ at time τ. If τ is even then $2\lfloor \tau/2 \rfloor + 1 = \tau + 1$; the synchronization occurs strictly after time τ, so we are guaranteed that no cell will ever enter state F before time τ in that case. The second point of Proposition 7.2 is therefore fulfilled. □

Two examples corresponding to the execution of $\mathscr{G}^1$ in the two cases of the parity of τ appear in Fig. 14, together with the two special cases $\tau = 2$ and $\tau = 3$. We now deal with the case τ even.

Proposition 7.3. *There is a $\mathscr{G}^2$ which*

(1) *synchronizes cells 1 to $\lfloor \tau/2 \rfloor$ at time τ if τ is even and $\tau \geqslant 2$;*

(2) *does not put any cell in state F before time τ if τ is odd ($\tau \geqslant 3$).*

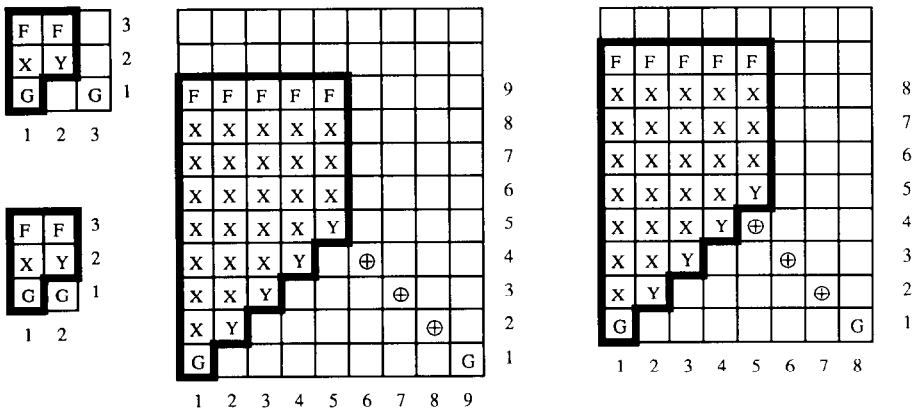

Fig. 14.

Proof. $\mathscr{G}^2$ is built exactly as $\mathscr{G}^1$ except that the start of the synchronization process is delayed by one instant. This way, in the case τ even, i.e. when the synchronization is achieved in time τ, the duration of the synchronization will be $\tau-1$, hence an odd number since τ is even. We can also see this the other way around. Since τ is even, that means that we will be able to cut the segment at time $\lfloor \tau/2 \rfloor + 1$ in two halves of length $\lfloor \tau/2 \rfloor$. To synchronize one of these halves with a minimal-time solution to the classical FSSP, it takes $2\lfloor \tau/2 \rfloor - 1$ time steps that is to say $\tau - 1$ steps (τ even$\Rightarrow \tau = 2\lfloor \tau/2 \rfloor$). If we want synchronization to occur at time τ we must therefore launch the synchronization process at time 2 instead of 1. To do that we arrange the transition function of $\mathscr{G}^2$ so as to put cell 1 at time 2 in a special new state G' which will play the same role as G at the next instants. The signal A and the right-end cell marking mechanism remain the same as in $\mathscr{G}^1$. This modification has the following consequences.

(1) If τ is even and $\tau \geqslant 4$ (the case $\tau = 2$ needs a special treatment, see below) then the right-end cell marking takes place in cell $\lfloor \tau/2 \rfloor$ at time $\lfloor \tau/2 \rfloor + 1$ (we leave the details of the proof of this fact to the reader), thus delimiting a segment of length $\lfloor \tau/2 \rfloor$. The cells of this segment will be synchronized at time $2\lfloor \tau/2 \rfloor - 1 + 1$ (this last $+1$ is due to the fact that the synchronization process began at instant 2 instead of 1), thus at time $2\lfloor \tau/2 \rfloor = \tau$ since τ is even. The first point of the proposition is thus fulfilled. Figure 15 gives an example of that situation.

(2) If τ is odd then the right-end cell marking takes place in cell $\lfloor \tau/2 \rfloor + 1$ at time $\lfloor \tau/2 \rfloor + 2$. The segment of length $\lfloor \tau/2 \rfloor + 1$ thus created synchronizes at time $2(\lfloor \tau/2 \rfloor + 1) - 1 + 1 = 2\lfloor \tau/2 \rfloor + 2 = \tau + 1$ since τ is odd. We are, therefore, sure that no cell ever enter state F before time τ in that case. This fulfils the second point of the proposition. Figure 16 shows an example of the execution of $\mathscr{G}^2$ in the case τ is odd.

We now treat the case $\tau = 2$. We then have $\lfloor \tau/2 \rfloor = 1$ so, to be consistent with the first point of Proposition 7.3, cell 1 should be in state F at time 2 and we should obtain what appears in Fig. 17. This means that cell 1 has to be put in state F if $\tau = 2$ and in state G' as it is said above only in the case $\tau \geqslant 3$. This is possible since, if $\tau = 2$ then

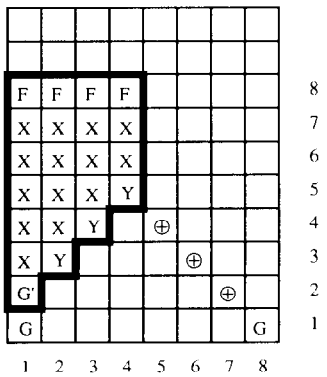

Fig. 15.

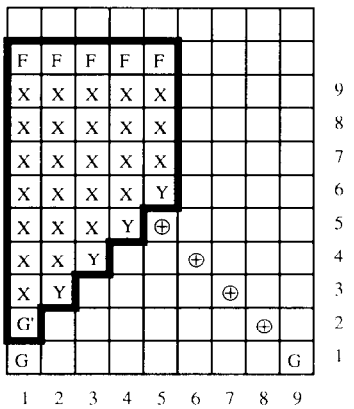

Fig. 16.

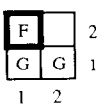

Fig. 17.

cell 1 at time 2 "sees" the G in the right-end cell at time 1, whereas it is not the case if $\tau \geqslant 3$. Cell 1 at time 2 can, therefore, "know" locally in what case we are and thus choose the appropriate state. □

We can now obtain the synchronization at time τ of the left half of the segment whatever the parity of τ. We shall obtain it by running $\mathscr{G}^1$ and $\mathscr{G}^2$ "in parallel" and

synchronizing when the first of the two synchronizes. This will work since we saw that, if τ is odd, then $\mathscr{G}^1$ is the first to synchronize, at time τ, whereas $\mathscr{G}^2$ waits until $\tau+1$ and that it is the contrary if τ is even.

Proposition 7.4. *There exists a* $\mathscr{G}^3$ *which*

(1) *synchronizes the cells* 1 *to* $\lfloor \tau/2 \rfloor + 1$ *at time* τ *if* τ *is odd;*

(2) *synchronizes the cells* 1 *to* $\lfloor \tau/2 \rfloor$ *at time* τ *if* τ *is even.*

Proof. We may obtain such a $\mathscr{G}^3$ through the superposition of $\mathscr{G}^1$ and $\mathscr{G}^2$. The set of states of $\mathscr{G}^3$ will be $T^3 = \{G, R, F\} \cup T^1 \times T^2$. At instant 1, the initial configuration remains as

$$\underbrace{GRR \ldots RG}_{\tau},$$

that is why G and $R \in T^3$. At any instant $t \geqslant 2$, the state of each cell is a pair (x, y) where x (y) is the state in which that cell would be at that time if $\mathscr{G}^1$ ($\mathscr{G}^2$) had been executed alone. To this rule we make two exceptions:

- If a cell had to be in state (R, R) according to the above rule then it is put in state R instead. This way, the state R remains a quiescent state.
- If a cell should be in state $(F, \ldots)$ or $(\ldots, F)$ then it is put in state F instead (that is why F belongs to T^3).

In these conditions, Propositions 7.2 and 7.3 imply that if τ is even then cells 1 to $\lfloor \tau/2 \rfloor$ will enter state F at time τ (due to $\mathscr{G}^2$), whereas if τ is odd then cells 1 to $\lfloor \tau/2 \rfloor + 1$ will enter state F at time τ (due to $\mathscr{G}^2$). On the other hand, we know that in the "wrong case" (τ even for $\mathscr{G}^1$ and τ odd for $\mathscr{G}^2$) no cell enters state F before time τ in the execution of $\mathscr{G}^1$ and $\mathscr{G}^2$, so it will also be the case for $\mathscr{G}^3$. We are, therefore, entitled to say that cells 1 to $\lfloor \tau/2 \rfloor$ if τ is even or 1 to $\lfloor \tau/2 \rfloor + 1$ if τ is odd are synchronized at time τ by $\mathscr{G}^3$. □

Figure 18 shows what happens when $\mathscr{G}^3$ is executed in the case τ even and τ odd. We may now extend our result to the other half of the segment.

Proposition 7.5. *There is a* $\mathscr{G}^4$ *which*

(1) *synchronizes cells* $\tau - \lfloor \tau/2 \rfloor + 1$ *to* τ *in time* τ *if* τ *is even;*

(2) *synchronizes cells* $\tau - \lfloor \tau/2 \rfloor$ *to* τ *in time* τ *if* τ *is odd.*

Proof. $\mathscr{G}^4$ may be obtained from $\mathscr{G}^3$ simply by permutating the roles of the two ends of the segment. To do this, we let $T^4 = T^3$ and for all x, y and $z \in T^4$:

$$\psi^4(x, y, z) = \psi^3(z, y, x),$$

$$\psi_l^4(x, y) = \psi_r^3(y, x),$$

$$\psi_r^4(x, y) = \psi_l^3(y, x). \qquad \square$$

We now have everything we need to obtain the minimal-time solution to the 2E-FSSP we were seeking.

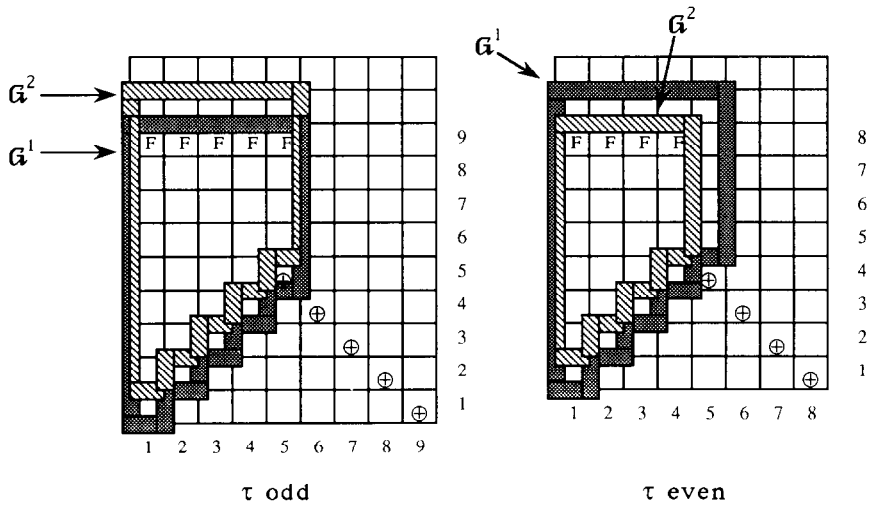

Fig. 18.

Theorem 7.6. *There is a solution $\mathcal{G}$ to the 2E-FSSP which works in minimal time.*

Proof. We obtain $\mathcal{G}$ from the superposition of $\mathcal{G}^3$ and $\mathcal{G}^4$ exactly as we obtained $\mathcal{G}^3$ from $\mathcal{G}^1$ and $\mathcal{G}^2$. Propositions 7.4 and 7.5 imply that if τ is even, then cells 1 to $\lfloor \tau/2 \rfloor$ and $\tau - \lfloor \tau/2 \rfloor + 1$ to τ are synchronized at time τ. But, since τ is even, we have $\tau = 2\lfloor \tau/2 \rfloor$, so $\lfloor \tau/2 \rfloor = \tau - \lfloor \tau/2 \rfloor$ and $\lfloor \tau/2 \rfloor + 1 = \tau - \lfloor \tau/2 \rfloor + 1$. This means that cells 1 to $\lfloor \tau/2 \rfloor$ plus $\tau - \lfloor \tau/2 \rfloor + 1$ to τ cover all cells from 1 to τ. All cells are synchronized at time τ. If τ is odd then cells 1 to $\lfloor \tau/2 \rfloor + 1$ and $\tau - \lfloor \tau/2 \rfloor$ are synchronized at time τ. Since τ is odd, it means that $\tau = 2\lfloor \tau/2 \rfloor + 1$; so, $\tau - \lfloor \tau/2 \rfloor = \lfloor \tau/2 \rfloor + 1$. The groups of cells 1 to $\lfloor \tau/2 \rfloor + 1$ and $\tau - \lfloor \tau/2 \rfloor$ cover, therefore, all cells 0 to τ (they are even overlapping). All cells are also synchronized in that case. □

Figure 19 shows the final aspect of the space–time diagram of our solution to the 2E-FSSP in the case τ is even and odd.

7.3. Building B^s

Using our minimal-time solution to the 2E-FSSP we shall build a B^s which puts cells $q-1, 2q-1, \ldots, q\lfloor n/q \rfloor - 1$ in state F at time $\lfloor n/q \rfloor$ and, hence, provides all cells 0 to $n+q-n \bmod q-1$ (we remember that $q\lfloor n/q \rfloor - 1 = n - n \bmod q - 1$) with the synchronizing information at time $\lfloor n/q \rfloor + 1$ as desired. As for the B^m of the preceding section, we only take here into account the cases of words u of length $n \geqslant q$ since in the case $n < q$ we have $\lfloor n/q \rfloor = 0$ and so the synchronization process has not even the time to perform one step. Again, we shall treat the case $n < q$ in Section 8.

Proposition 7.7. *There is an automaton $B^s = (Q^s, q, g^s)$ which puts cells $q-1, 2q-1, \ldots, q\lfloor n/q \rfloor - 1$ in state F at time $\lfloor n/q \rfloor$ and does not put any other cell in state F before that time, when given the initial configuration $c_{\mathcal{G}}[u]$ corresponding to a word u of length $n \geqslant q$.*

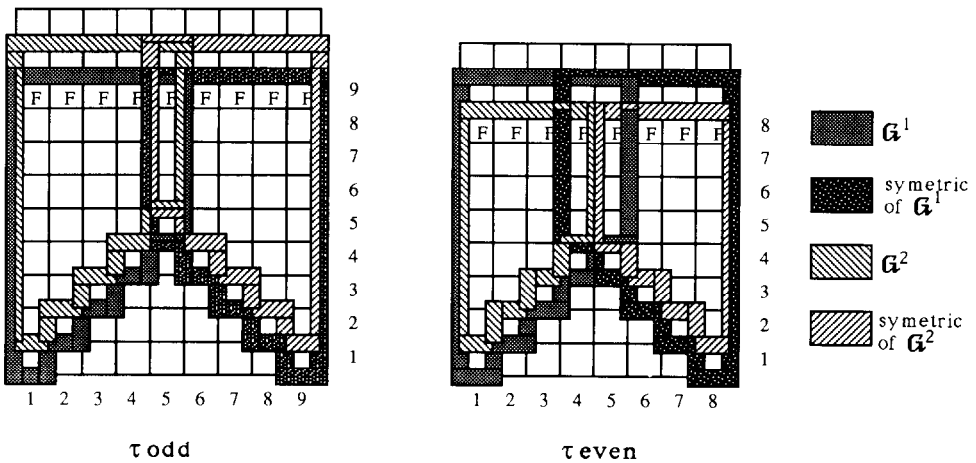

Fig. 19.

Proof. Basically, if $n \geqslant 2q$, then B^s simulates the execution of $\mathcal{G}$, our solution to the 2E-FSSP, on cells $q-1, 2q-1, \ldots, q\lfloor n/q \rfloor - 1$ (since $n \geqslant 2q$, we have at least two cells in the simulated segment as required for the 2E-FSSP). We shall deal with the special case $q \leqslant n < 2q$ at the end of the proof.

We let $Q^s = U \cup \{(q_0, \ldots, q_0)_\alpha, (\lambda, \ldots, \lambda)_\alpha\} \cup T$, the set $U \cup \{(q_0, \ldots, q_0)_\alpha, (\lambda, \ldots, \lambda)_\alpha\}$ being the alphabet of the initial configuration $c_\mathcal{G}[u]$ and T, the set of states of $\mathcal{G}$. Let us suppose that we have $n \geqslant 2q$. At instant 0, the cells of the line contain the letters of the initial configuration $c_\mathcal{G}[u]$. At time 1, the cells 0 to $n-1$ will be put in states belonging to T according to an initialization process described below. At any time, the negative cells remain in state $(\lambda, \ldots, \lambda)_\alpha$ and the cells $n, n+1, \ldots$ in state $(q_0, \ldots, q_0)_\alpha$. After time 1, we must simulate the behaviour of the transition function of $\mathcal{G}$ on cells $q-1$, $2q-1, \ldots, q\lfloor n/q \rfloor - 1$. We shall do it in the following way.

- Two consecutive cells of the sequence $q-1, 2q-1, \ldots, q\lfloor n/q \rfloor - 1$ are distant of q positions. Since we want to simulate on them the behaviour of an automaton of range 1, it means that the state of the nonend cells (cells $2q-1, \ldots, q(\lfloor n/q \rfloor - 1) - 1$) must be determined according to that of themselves and that of their two neighbours lying on both sides at distance q: We let, for all $a, b, c \in T$

$$g^s(\underbrace{a, \ldots}_{q \text{ elements}}, \underbrace{b, \ldots}_{q \text{ elements}}, c) = \psi(a, b, c),$$

where ψ is the transition function of $\mathcal{G}$.

- Since $n \geqslant 2q$, we have two distinct end cells: $q-1$ and $q\lfloor n/q \rfloor - 1$. Cell $q-1$ always sees one cell in state $(\lambda, \ldots, \lambda)_\alpha$ on its left and cell $q\lfloor n/q \rfloor - 1$ ($= n - n \bmod q - 1$) sees

at least one cell in state $(q_0,\dots,q_0)_x$ on its right. These facts, therefore, allow these cells to be identified locally. We let, for all $a,b\in T$:

$$g^s(\underbrace{(\lambda,\dots,\lambda)_x}_{q\text{ elements}},\dots,\underbrace{a,\dots,b}_{q\text{ elements}})=\psi_l(a,b)$$

and

$$g^s(\underbrace{a,\dots,b}_{q\text{ elements}},x_1,x_2,\dots,x_q)=\psi_r(a,b),$$

if one of the x_i at least is a $(q_0,\dots,q_0)_x$.

This way, we obtain the desired result on cells $q-1,2q-1,\dots,q\lfloor n/q\rfloor-1$. We now deal with the initialization of cells 1 to $n-1$ at time 1. Ideally, we would like to put cells $q-1$ and $q\lfloor n/q\rfloor-1$ in state G, while putting all the other cells in state R. This way, the synchronizing process would take place on cells $q-1,2q-1,\dots,q\lfloor n/q\rfloor-1$, synchronizing properly at time $\lfloor n/q\rfloor$, whereas the other cells would remain in state R all the time. However, this is not exactly possible, since, unlike cell $q-1$ (which may be locally identified as being the rightmost cell to see the left extremity of the word u), the cell $q\lfloor n/q\rfloor-1$ $(=n-n \bmod q-1)$ cannot be identified locally at time 1 since we do not know $n \bmod q$ at that time. To overcome this problem we shall do the following things:

- We put cell $q-1$ in state G at time 1. We may do this since cell $q-1$ is locally identifiable as we said above.
- We put *all* cells $n-q$ to $n-1$ in state G at time 1. This is clearly possible since these cells are those "which see the right end of the word u on their right". We are sure that one of these cells is cell $q\lfloor n/q\rfloor-1$, which is, therefore, put in state G. On the other hand, no other cell of the sequence $q-1,2q-1,\dots,q\lfloor n/q\rfloor-1$ is put in state G due to that operation since $q(\lfloor n/q\rfloor-1)-1=n-n \bmod q-1-q$ is strictly inferior to $n-q$ and $q-1<n-q$ (since $n\geqslant 2q$).
- All the other cells in the range 0 to $n-1$ are put in state R at time 1.

This way, B^s simulates q synchronization processes which take place on q segments contained in the interleaved sequences of cells $s(i)=(n-qj+i)_{1\leqslant j\leqslant\lfloor n/q\rfloor}$, with $1\leqslant i\leqslant q$. The right-end cell of all these segments is in state G at time 1 but only one, the segment contained in the sequence $s(q-n \bmod q-1)$ which corresponds to the cells $q-1$, $2q-1,\dots,q\lfloor n/q\rfloor-1$, has also its left-end cell in state G, whereas all others have their left-end cells in state R. Therefore, only this segment will be synchronized at time $\lfloor n/q\rfloor$, thus yielding the desired result, while the $q-1$ others will only contain one half of the synchronization process (since the half which should have appeared from the left-end cell is missing) and will, therefore, not be synchronized at any time between 1 and $\lfloor n/q\rfloor$ as shown in Fig. 20.

Let us now deal with case $n\leqslant q<2q$. We have $\lfloor n/q\rfloor=1$, so the sequence of cells $q-1,2q-1,\dots,q\lfloor n/q\rfloor-1$ is reduced to the unique cell $q-1$. In order to maintain consistency with the case $n\geqslant 2q$ this cell must be put in state F at time 1. Cell $q-1$

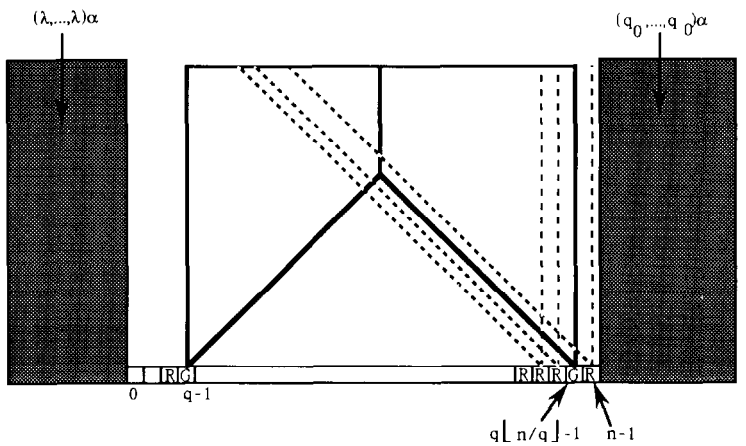

Fig. 20.

must, therefore, be able to choose at time 1 between state G, for the case $n \geqslant 2q$, and state F for the case $q \leqslant n < 2q$. Fortunately, in the case $q \leqslant n < 2q$ cell $q-1$ sees at time 1 the right extremity of the word u, whereas this is not the case if $n \geqslant 2q$. So, cell $q-1$ can know locally at time 1 if we are in case $n \geqslant q$ or $q \leqslant n < 2q$. We can, therefore, arrange g^s so as to choose the appropriate state. □

8. The speed-up theorem—strong form

In the preceding four sections, we have accumulated all the necessary components to build an automaton that will fulfil the requirements of the strong form of the linear speed-up theorem for CAs:

(1) An automaton which computes k times faster on grouped configurations and saves the end states: the B^* of Section 4.

(2) An automaton which yields an "almost grouped" ($\gamma_\alpha^{n \bmod q}$-grouped) configuration at time $\lfloor n/q \rfloor$ from the initial configuration $c_{\mathscr{A}}[u]$: the B^g of Section 5.

(3) An automaton which provides all cells 0 to $n+q-n \bmod q-1$ with the value of $n \bmod q$ at time $\lfloor n/q \rfloor + 1$ (n is the length of the input word u contained in the initial configuration $c_{\mathscr{A}}[u]$): the B^m of Section 6.

(4) An automaton which synchronizes all cells 0 to $n+q-n \bmod q-1$ at time $\lfloor n/q \rfloor + 1$: the B^s of Section 6.

From B^g, B^m and B^s we shall finally obtain the announced grouping automaton B^G which will yield at time $\lfloor n/q \rfloor + 1$ the completely grouped (γ_α^0-grouped) configuration

$$\sigma_{\lfloor n/q \rfloor + 1} = \gamma_\alpha^0(G_f^{(\lfloor n/q \rfloor + 1)}(c_{\mathscr{A}}[u])),$$

starting from the initial configuration $c_{\mathscr{A}}[u] = \gamma_\alpha^n(c_{\mathscr{A}}[u])$.

Lemma 8.1. *Let* $\mathscr{A}=(U,\lambda,q_0,Q_a,Q_r,A)$ *with* $A=(Q,p,f)$ *be a recognizer CA which recognizes the language* $L\subset U^*$ *in time* $t(n)$. *For all* $\alpha\geqslant 2$ *and all* $q\geqslant p$ *there is an automaton* B^G *of range* q *which, fed with the initial configuration* $c_{\mathscr{C}}[u]$ *corresponding to any word* $u\in U^*$, *yields the configuration*

$$\sigma_{\lfloor n/q\rfloor+1}=\gamma_\alpha^0(G_f^{(\lfloor n/q\rfloor+1)}(c_{\mathscr{A}}[u]))$$

at time $\lfloor n/q\rfloor+1$.

Proof. For initial configurations $c_{\mathscr{C}}[u]$ corresponding to words u of length $n\geqslant q$, we shall have B^G simulating "in parallel" the execution of B^g, B^m and B^s (which are only designed to work with input words of length $n\geqslant q$) from time 1 to time $\lfloor n/q\rfloor$. At time $\lfloor n/q\rfloor+1$, the configuration $\sigma_{\lfloor n/q\rfloor+1}$ will be obtained through a special procedure described below. We shall treat the case $n<q$ at the end of the proof. Let us first suppose that we have $n\geqslant q$ and, hence, $\lfloor n/q\rfloor\geqslant 1$.

(A) We shall first deal with the behaviour of B^G from time 1 to time $\lfloor n/q\rfloor$. The state of any cell at any time in that period will be a triplet (x,y,z), where x (y or z) is the state in which this cell would have been at that time if B^g (B^m or B^s) had been executed alone on the initial configuration $c_{\mathscr{C}}[u]$. We have $x\in Q^g$, $y\in Q^m$ and $z\in Q^s$. From the space–time diagram of B^g, B^m and B^s (see Figs. 5, 6 and 20) we see that in all three cases, all the negative cells remain in state $(\lambda,\ldots,\lambda)_\alpha$ all the time, while the positive cells after cell n remain in state $(q_0,\ldots,q_0)_\alpha$ all the time. Instead of putting these cells in B^G in states $((\lambda,\ldots,\lambda)_\alpha, (\lambda,\ldots,\lambda)_\alpha, (\lambda,\ldots,\lambda)_\alpha)$ or $((q_0,\ldots,q_0)_\alpha, (q_0,\ldots,q_0)_\alpha, (q_0,\ldots,q_0)_\alpha)$, respectively, as we should do according to the above superposition principle, we shall simply put them in states $(\lambda,\ldots,\lambda)_\alpha$ and $(q_0,\ldots,q_0)_\alpha$ instead. This way, $(\lambda,\ldots,\lambda)_\alpha$ will still have the properties of a border state and $(q_0,\ldots,q_0)_\alpha$ that of a quiescent state which is important since they will play these roles in our speeded-up automaton $\mathscr{C}$. Intuitively, all this gives to the space–time diagram of B^G the "layered structure", which appears in Fig. 21. It is clear that we can arrange g^G, the transition function of B^G, so as to yield such a diagram.

(B) At time $\lfloor n/q\rfloor+1$ we want to obtain the configuration

$$\sigma_{\lfloor n/q\rfloor+1}=\gamma_\alpha^0(G_f^{(\lfloor n/q\rfloor+1)}(c_{\mathscr{A}}[u])).$$

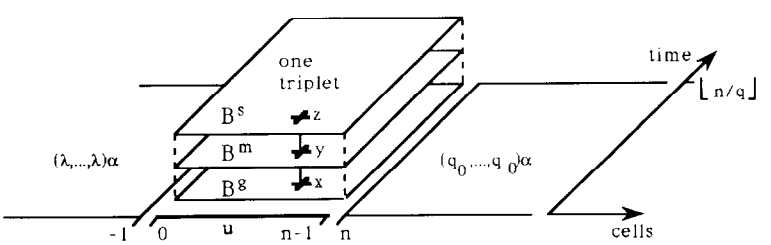

Fig. 21.

At time $\lfloor n/q \rfloor$ we have, contained in the layer of B^{g}, the configuration

$$\sigma_{\lfloor n/q \rfloor} = \gamma_\alpha^{n \bmod q}(G_f^{(\lfloor n/q \rfloor)}(c_{\mathscr{A}}[u])).$$

We showed at the end of Section 5 that there exists an automaton which can pass in one step between these two configurations, provided that at least all the active cells (those which are not in state $(\lambda,\ldots,\lambda)_\alpha$ or $(q_0,\ldots,q_0)_\alpha$) of $\sigma_{\lfloor n/q \rfloor+1}$ are provided with the value of $n \bmod q$ and with information which indicates to them that they are active. We also saw that the active cells of the configuration $\sigma_{\lfloor n/q \rfloor+1}$ are the cells 0 to $n+q-n \bmod q-1$. Due to the presence of a layer containing B^{s}, these cells are synchronized at time $\lfloor n/q \rfloor+1$. This means that they see something (the F contained in the cells $q-1, 2q-1,\ldots, q\lfloor n/q \rfloor-1$ at time $\lfloor n/q \rfloor$) that no other cells saw at any time before. We are, therefore, free to give them any behaviour we want. Besides that, the synchronization information may be seen as the information "you are active" and these cells are also provided with the value of $n \bmod q$ through the layer of B^{m}. We can, therefore, arrange B^{m} so as to behave on cells 0 to $n+q-n \bmod q-1$ as the transition function of the automaton alluded to above and which passes from $\sigma_{\lfloor n/q \rfloor}$ to $\sigma_{\lfloor n/q \rfloor+1}$. All the negative cells and those after cell $n+q-n \bmod q-1$ do not see the synchronization, so they do not know that they are at time $\lfloor n/q \rfloor+1$. g^{G} will, therefore, act on them as at the previous instants, putting them in states $(\lambda,\ldots,\lambda)_\alpha$ and $(q_0,\ldots,q_0)_\alpha$, respectively. Since we showed that these cells are not active cells of $\sigma_{\lfloor n/q \rfloor+1}$, these values are adequate. In this way, we indeed obtain the configuration $\sigma_{\lfloor n/q \rfloor+1}$ at time $\lfloor n/q \rfloor+1$.

Let us now deal with case $n<q$. We have $\lfloor n/q \rfloor=0$, so $\lfloor n/q \rfloor+1=1$. We have to yield the configuration $\sigma_{\lfloor n/q \rfloor+1}$ at time 1, in one step. In case $n<q$, we have $n \bmod q=n$, so $n+q-n \bmod q-1=q-1$ and, hence, all cells 0 to $n+q-n \bmod q-1$ (the active cells of $\sigma_{\lfloor n/q \rfloor+1}$) see both ends of the word u at time 1 and, therefore, know that it is of length $n<q$. It is clear that in the case $n \geqslant q$, no cell is ever in that situation. The cells 0 to $n+q-n \bmod q-1$ can, therefore, decide locally if they are in the case $n<q$ or $n \geqslant q$. Since they all see the whole of u, they can clearly be put in the appropriate state to yield the letters of $\sigma_{\lfloor n/q \rfloor+1}$ corresponding to their positions. On the other hand, the negative cells and those after cell $n+q-n \bmod q-1$ are in the same situation in the case $n<q$ as in any other case. g^{G} puts them, therefore, in states $(\lambda,\ldots,\lambda)_\alpha$ and $(q_0,\ldots,q_0)_\alpha$, which are adequate since we know that they are not active cells of $\sigma_{\lfloor n/q \rfloor+1}$. □

We have represented in Fig. 22 the aspect of the layer of B^{g} in the diagram of B^{G} plus the configuration $\sigma_{\lfloor n/q \rfloor+1}$ in the same situation as the example of Fig. 5. We can now state a first version of the strong form of the linear speed-up theorem for CAs.

Theorem 8.2. *Let $\mathscr{A}=(U,\lambda,q_0,Q_{\mathrm{a}},Q_{\mathrm{r}},A)$, with $A=(Q,p,f)$, be a recognizer CA which recognizes the language $L \subset U^*$ in time $t(n)$. For any $k \geqslant 1$ and any $q \geqslant p$ there is another*

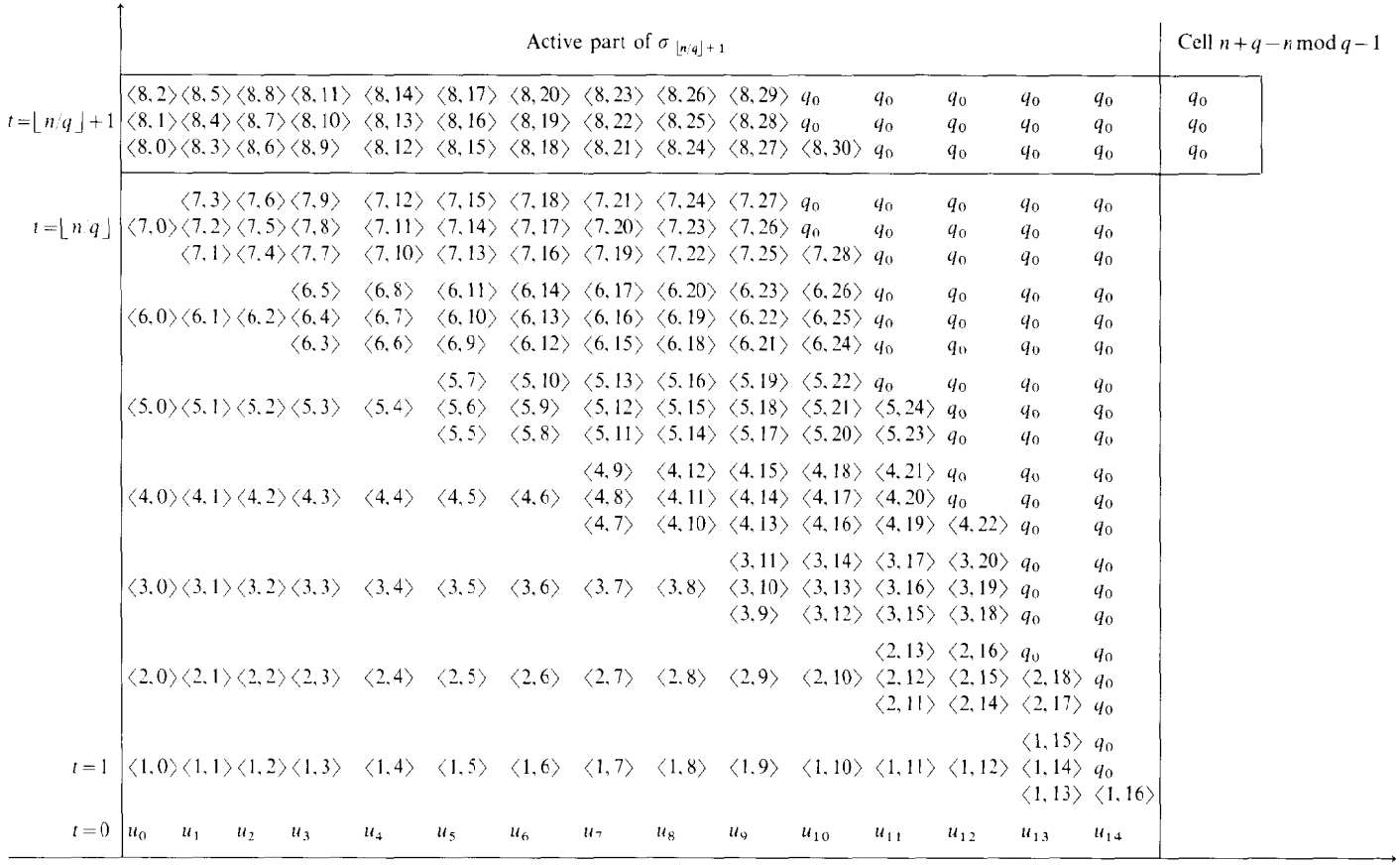
Active part of $\sigma_{\lfloor n/q \rfloor + 1}$
Cell $n+q-n \bmod q-1$
$t=\lfloor n/q \rfloor+1$
$t=\lfloor n/q \rfloor$
$t=1$
$t=0$
$\langle 8,2\rangle\langle 8,5\rangle\langle 8,8\rangle\langle 8,11\rangle\langle 8,14\rangle\langle 8,17\rangle\langle 8,20\rangle\langle 8,23\rangle\langle 8,26\rangle\langle 8,29\rangle q_0\ q_0\ q_0\ q_0\ q_0$ q_0
$\langle 8,1\rangle\langle 8,4\rangle\langle 8,7\rangle\langle 8,10\rangle\langle 8,13\rangle\langle 8,16\rangle\langle 8,19\rangle\langle 8,22\rangle\langle 8,25\rangle\langle 8,28\rangle q_0\ q_0\ q_0\ q_0\ q_0$ q_0
$\langle 8,0\rangle\langle 8,3\rangle\langle 8,6\rangle\langle 8,9\rangle\langle 8,12\rangle\langle 8,15\rangle\langle 8,18\rangle\langle 8,21\rangle\langle 8,24\rangle\langle 8,27\rangle\langle 8,30\rangle q_0\ q_0\ q_0\ q_0$ q_0
$\langle 7,3\rangle\langle 7,6\rangle\langle 7,9\rangle\langle 7,12\rangle\langle 7,15\rangle\langle 7,18\rangle\langle 7,21\rangle\langle 7,24\rangle\langle 7,27\rangle q_0\ q_0\ q_0\ q_0\ q_0$
$\langle 7,0\rangle\langle 7,2\rangle\langle 7,5\rangle\langle 7,8\rangle\langle 7,11\rangle\langle 7,14\rangle\langle 7,17\rangle\langle 7,20\rangle\langle 7,23\rangle\langle 7,26\rangle q_0\ q_0\ q_0\ q_0\ q_0$
$\langle 7,1\rangle\langle 7,4\rangle\langle 7,7\rangle\langle 7,10\rangle\langle 7,13\rangle\langle 7,16\rangle\langle 7,19\rangle\langle 7,22\rangle\langle 7,25\rangle\langle 7,28\rangle q_0\ q_0\ q_0\ q_0$
$\langle 6,5\rangle\langle 6,8\rangle\langle 6,11\rangle\langle 6,14\rangle\langle 6,17\rangle\langle 6,20\rangle\langle 6,23\rangle\langle 6,26\rangle q_0\ q_0\ q_0\ q_0$
$\langle 6,0\rangle\langle 6,1\rangle\langle 6,2\rangle\langle 6,4\rangle\langle 6,7\rangle\langle 6,10\rangle\langle 6,13\rangle\langle 6,16\rangle\langle 6,19\rangle\langle 6,22\rangle\langle 6,25\rangle q_0\ q_0\ q_0\ q_0$
$\langle 6,3\rangle\langle 6,6\rangle\langle 6,9\rangle\langle 6,12\rangle\langle 6,15\rangle\langle 6,18\rangle\langle 6,21\rangle\langle 6,24\rangle q_0\ q_0\ q_0\ q_0$
$\langle 5,7\rangle\langle 5,10\rangle\langle 5,13\rangle\langle 5,16\rangle\langle 5,19\rangle\langle 5,22\rangle q_0\ q_0\ q_0\ q_0$
$\langle 5,0\rangle\langle 5,1\rangle\langle 5,2\rangle\langle 5,3\rangle\langle 5,4\rangle\langle 5,6\rangle\langle 5,9\rangle\langle 5,12\rangle\langle 5,15\rangle\langle 5,18\rangle\langle 5,21\rangle\langle 5,24\rangle q_0\ q_0\ q_0$
$\langle 5,5\rangle\langle 5,8\rangle\langle 5,11\rangle\langle 5,14\rangle\langle 5,17\rangle\langle 5,20\rangle\langle 5,23\rangle q_0\ q_0\ q_0$
$\langle 4,9\rangle\langle 4,12\rangle\langle 4,15\rangle\langle 4,18\rangle\langle 4,21\rangle q_0\ q_0\ q_0$
$\langle 4,0\rangle\langle 4,1\rangle\langle 4,2\rangle\langle 4,3\rangle\langle 4,4\rangle\langle 4,5\rangle\langle 4,6\rangle\langle 4,8\rangle\langle 4,11\rangle\langle 4,14\rangle\langle 4,17\rangle\langle 4,20\rangle q_0\ q_0\ q_0$
$\langle 4,7\rangle\langle 4,10\rangle\langle 4,13\rangle\langle 4,16\rangle\langle 4,19\rangle\langle 4,22\rangle q_0\ q_0$
$\langle 3,11\rangle\langle 3,14\rangle\langle 3,17\rangle\langle 3,20\rangle q_0\ q_0$
$\langle 3,0\rangle\langle 3,1\rangle\langle 3,2\rangle\langle 3,3\rangle\langle 3,4\rangle\langle 3,5\rangle\langle 3,6\rangle\langle 3,7\rangle\langle 3,8\rangle\langle 3,10\rangle\langle 3,13\rangle\langle 3,16\rangle\langle 3,19\rangle q_0\ q_0$
$\langle 3,9\rangle\langle 3,12\rangle\langle 3,15\rangle\langle 3,18\rangle q_0\ q_0$
$\langle 2,13\rangle\langle 2,16\rangle q_0\ q_0$
$\langle 2,0\rangle\langle 2,1\rangle\langle 2,2\rangle\langle 2,3\rangle\langle 2,4\rangle\langle 2,5\rangle\langle 2,6\rangle\langle 2,7\rangle\langle 2,8\rangle\langle 2,9\rangle\langle 2,10\rangle\langle 2,12\rangle\langle 2,15\rangle\langle 2,18\rangle q_0$
$\langle 2,11\rangle\langle 2,14\rangle\langle 2,17\rangle q_0$
$\langle 1,15\rangle q_0$
$\langle 1,0\rangle\langle 1,1\rangle\langle 1,2\rangle\langle 1,3\rangle\langle 1,4\rangle\langle 1,5\rangle\langle 1,6\rangle\langle 1,7\rangle\langle 1,8\rangle\langle 1,9\rangle\langle 1,10\rangle\langle 1,11\rangle\langle 1,12\rangle\langle 1,14\rangle q_0$
$\langle 1,13\rangle\langle 1,16\rangle$
$u_0\ u_1\ u_2\ u_3\ u_4\ u_5\ u_6\ u_7\ u_8\ u_9\ u_{10}\ u_{11}\ u_{12}\ u_{13}\ u_{14}$
cell 0
The input word u
cell $n-1$

Fig. 22.

recognizer CA $\mathscr{C}=(U,(\lambda,\ldots,\lambda)_\alpha,(q_0,\ldots,q_0)_\alpha,R_a,R_r,C)$, *with* $C=(R,q,h)$, *which recognizes the language* L *in time*

$$\lfloor n/q\rfloor+1+\left\lceil\frac{t(n)-\lfloor n/q\rfloor-1}{k}\right\rceil$$

if $t(n)\geqslant\lfloor n/q\rfloor+1$ *and* $t(n)$ *in the opposite case.*

Proof. Let us take an $\alpha\geqslant 2$ such that $q\alpha\geqslant kp$ (the necessary condition for the existence of a speeded-up automaton B^*). Let us consider the grouping automaton B^G and the speeded-up automaton B^* corresponding to that α. We shall obtain $\mathscr{C}$ from the composition of B^G and B^*, i.e. we are going to show that we can define C so as to behave like B^G from the initial configuration

$$c_{\mathscr{C}}[u]=\ldots(\lambda,\ldots,\lambda)_\alpha u_0u_1\ldots u_{n-1}(q_0,\ldots,q_0)_\alpha(q_0,\ldots,q_0)_\alpha\ldots$$

for any word $u\in U^*$, until time $\lfloor n/q\rfloor+1$ and then like B^* until an accepting or rejecting state appears. The configurations on which C has to operate are of one of the three following types:

(1) the configuration $c_{\mathscr{C}}[u]$,

(2) the configurations of B^G from time 1 to time $\lfloor n/q\rfloor$,

(3) the configuration $\sigma_{\lfloor n/q\rfloor+1}$ and the configurations of B^* which are all γ_α^0-grouped configurations belonging to $(Q^\alpha)^{\mathbf{Z}}$.

On types (1) and (2), C must behave like B^G and as B^* on type (3). All configurations of the three types have the following common structure: $(\lambda,\ldots,\lambda)_\alpha$ states until cell -1, then a finite number of *active cells* in states belonging to a certain alphabet (depending on the type), and then the $(q_0,\ldots,q_0)_\alpha$ states. This alphabet is U for type (1), the set of triplets of states of B^g, B^m and B^s for type (2) and $Q^\alpha-\{(\lambda,\ldots,\lambda)_\alpha,(q_0,\ldots,q_0)_\alpha\}$ for type (3). Whatever the type, when h, the transition function of C sees only $(\lambda,\ldots,\lambda)_\alpha$ or $(q_0,\ldots,q_0)_\alpha$ states, it has to (and may) behave the same way, yielding $(\lambda,\ldots,\lambda)_\alpha$ or $(q_0,\ldots,q_0)_\alpha$, respectively. The key aspect of this proof is that the three alphabets of the active parts of the configurations of the three types are *disjoints*; therefore, h knows that

- when it sees at least one letter of U or of the set of triplets then it is working on a configuration of type (1) or (2) and, therefore, that it has to behave like g^G, the transition function of B^G;
- when it sees at least one letter of $Q^\alpha-\{(\lambda,\ldots,\lambda)_\alpha,(q_0,\ldots,q_0)_\alpha\}$ then it knows that it is working on a configuration of type (3) and that it has, therefore, to behave like g^* the transition function of B^*.

In that sense, we may say that the behaviour of B^G and B^* are *independent* and that they can, therefore, be adopted by an unique CA, composition of B^G and B^*. This property of independence is obtained thanks to the presence of the synchronizing process of B^s, which enables us to pass between instants $\lfloor n/q\rfloor$ and $\lfloor n/q\rfloor+1$ from the alphabet of triplets to $Q^\alpha-\{(\lambda,\ldots,\lambda)_\alpha,(q_0,\ldots,q_0)_\alpha\}$: two *disjoint* alphabets.

We now take $R_a = Q_a \times Q^{\alpha-1} \cup Q_a \times Q^m \times Q^s$ and $R_r = Q_r \times Q^{\alpha-1} \cup Q_r \times Q^m \times Q^s$. If $t(n) < \lfloor n/q \rfloor + 1$ then an accepting (rejecting) state appears in the process of B^G. The state in which cell 0 is at that moment is in $Q_a \times Q^m \times Q^s$ ($Q_r \times Q^m \times Q^s$). The recognition time remains $t(n)$. If $t(n) \geqslant \lfloor n/q \rfloor + 1$ then $t(n) - \lfloor n/q \rfloor - 1$ steps of the computation of $\mathscr{A}$ remain to be done after time $\lfloor n/q \rfloor + 1$. This duration is reduced to

$$\left\lceil \frac{t(n) - \lfloor n/q \rfloor - 1}{k} \right\rceil$$

by B^*, thus yielding a total recognition time of

$$\lfloor n/q \rfloor + 1 + \left\lceil \frac{t(n) - \lfloor n/q \rfloor - 1}{k} \right\rceil.$$

At that instant, cell 0 will be in a state belonging to the set $Q_a \times Q^{\alpha-1}$ or $Q_r \times Q^{\alpha-1}$. □

Remark. Let us suppose that the number of states of $\mathscr{A}$ is s. The number of states of B^g is $s_g = O(s^\alpha)$, that of B^m is $s_m = O(q)$ (since Q^m contains $\{0, 1, 2, \ldots, q-1\}$), that of B^s is a constant s_s and that of B^* is $s_* = O(s^\alpha)$. Therefore, the number of state of $\mathscr{C}$ is $s_g \times s_m \times s_s + s_* = O(qs^\alpha)$. If we work only with automata of range 1 then we have $q = 1$ and $\alpha = k$ since we must have $q\alpha \geqslant kp$ and $q = p = 1$. The number of states of $\mathscr{C}$ in these conditions is, therefore, $O(s^k)$, as announced in the Introduction.

In the formulation of the above theorem, there still was the restriction $q \geqslant p$ inherited from B^g (see Section 5). We now propose a second version of the theorem, in which we get rid of that condition but at the price of some loss in time efficiency.

Theorem 8.3. *Let $\mathscr{A}$ be a recognizer CA of range p which recognizes the language L in time $t(n)$. For any $k \geqslant 1$ and any $q' \geqslant 1$ there is another recognizer CA $\mathscr{C}'$ of range q' which recognizes the language L in time*

$$\lfloor n/q' \rfloor + 1 + \left\lceil \frac{\lceil p/q' \rceil t(n) - \lfloor n/q' \rfloor - 1}{k} \right\rceil,$$

if $\lceil p/q' \rceil t(n) \geqslant \lfloor n/q' \rfloor + 1$ and $\lceil p/q' \rceil t(n)$ in the opposite case.

Proof. From an $\mathscr{A}$ of range p we can build an $\mathscr{A}'$ of range $q' < p$ which simulates $\mathscr{A}$ in the following way: The state of each cell is communicated to as many neighbouring cells as possible. After $\lceil p/q' \rceil$ time steps of this process, each cell knows enough information to perform on step of $\mathscr{A}$. $\mathscr{A}'$ will, therefore, recognize the same language as $\mathscr{A}$ in time $\lceil p/q' \rceil t(n)$. If $q' \geqslant p$ then we take $\mathscr{A}' = \mathscr{A}$, and this way we have also an $\mathscr{A}'$ working in time $\lceil p/q' \rceil t(n)$ since in that case, $\lceil p/q' \rceil = 1$. If we now accelerate $\mathscr{A}'$ with Theorem 8.2 we may obtain a $\mathscr{C}'$ of range q' which recognizes the language L in time as stated above. □

Remark. If $q' \geqslant q$ we have $\lceil p/q' \rceil = 1$ we obtain, with Theorem 8.3, the same result as with Theorem 8.2.

9. Conclusion

In the proof of Theorem 8.2 we obtained $\mathscr{C}$ through a so-called composition of B^g and B^*. We may generalize this notion in the following way. Let us assume that we want to compose two *computing* CAs $\mathscr{N}$ and $\mathscr{M}$ which compute the recursive functions $\varphi: U^* \to V^*$ and $\psi: V^* \to W^*$ in time $t(n)$ and $s(n)$, respectively. We mean by this that $\mathscr{M}$ ($\mathscr{N}$) started with an initial configuration[9] $c_{\mathscr{M}}[u]$ for $u \in U^*$ ($c_{\mathscr{N}}[v]$ for $v \in V^*$) yields the configuration $c_{\mathscr{M}}[\varphi(u)]$ ($c_{\mathscr{N}}[\psi(u)]$) after $t(n)$ ($s(n)$) time steps. We assume that $\mathscr{M}$ and $\mathscr{N}$ have the same border state and the same quiescent state, so that $c_{\mathscr{M}}[v] = c_{\mathscr{N}}[v]$ for any $v \in V^*$: any output configuration of $\mathscr{M}$ may be used as an input configuration of $\mathscr{N}$. Composing $\mathscr{M}$ and $\mathscr{N}$ means building an automaton $\mathscr{P}$ which will work like $\mathscr{M}$ until time $t(n)$ and then "switch" to work like $\mathscr{N}$. To achieve this we must render the computation of $\mathscr{M}$ and $\mathscr{N}$ *independent*, i.e. what we require of the transition function of $\mathscr{P}$, in order to perform the computation of $\mathscr{M}$, must not come into conflict with what we require of it in order to perform the computation of $\mathscr{N}$. A sufficient condition to achieve that goal is to represent the computation of $\mathscr{M}$ and $\mathscr{N}$ on two disjoint alphabets, shifting from one to the other from time $t(n)-1$ to $t(n)$. To make sure of that we may put a special marker on all (nonquiescent) cells participating in the computation of $\mathscr{M}$ until time $t(n)-1$. At time $t(n)$, the marking must be removed from all the cells which should have borne it, had the computation of $\mathscr{M}$ continued; i.e. from all cells containing the letters of $\varphi(u)$. In other words, all cells 0 to $|\varphi(u)|-1$ must be synchronized at that time. If we are able to do that then we may build $\mathscr{P} = \mathscr{N} \circ \mathscr{M}$.

The problem of Composing two computing CAs is, therefore, reduced to something we may call as *space–time constructibility* on cellular automata: A couple of functions $(\tau(n), \chi(n))$ is said to be space–time constructible if there is a CA capable of synchronizing $\chi(n)$ cells in time $\tau(n)$ for all n. We shall be able to compose $\mathscr{M}$ and $\mathscr{N}$ if the couple $(t(n), l(n))$ with $l(n) = \max\{|\varphi(u)|, |u| = n\}$ is space–time constructible on CA.

Classical solutions to the FSSP provide the space–time constructibility of the following couples of functions: (n, n) (minimal-time 2E-FSSP, used here), $(2n-1, n)$ (minimal-time FSSP), $(3n, n)$ (Minsky's early solution to the FSSP). Two approaches have been used to obtain new couples. One may try completely new ideas of synchronization processes as Moore did in [10] for the couple (n^2, n). Some new results in this direction will be presented in [8]. One may also use minimal-time solutions to the FSSP (or 2E-FSSP) launched at an appropriate instant by a "triggering" signal. The Erathosthene sieve cellular automaton of Fisher in [5] contains a signal which reaches cell k at time approximately k^2. Using a similar signal we could launch a minimal-time FSSP solution from cell n at time n^2, propagating leftwards, thus obtaining the space–time constructibility of the couple $(n^2 + 2n - 1, n)$. Signals of

[9] We recall that the configuration $c_{\mathscr{A}}[u]$ corresponding to the word u for the automaton $\mathscr{A}$ is obtained by assigning to negative cells the border state of $\mathscr{A}$, to positive cells 0 to $|u|-1$ the letters of u, and to all the next ones the quiescent state of $\mathscr{A}$.

various kinds and speeds may be found in many papers on CAs. Among them, Culik and Choffrut in [4] provide an exponential signal, Terrier in [12] another exponential, a square root and a logarithmic one. However, these two problems—signal building and space–time constructibility—have not yet received the complete and detailed treatment they deserve.

In the future, it may be desirable to extend to *computing* CAs our result, which only applies for the present to *recognizer* CAs. To achieve this will require a great deal more work in the area of CA composition and space–time CA-constructible functions, since such a result would involve the composition of an automaton similar to our $\mathscr{C}$ with an *ungrouping* automaton. To do this we will have to be able to synchronize at the end of the computation of a $\mathscr{C}$ whose execution time is arbitrary. If we do not want to have to reduce too much the class of admitted $\mathscr{C}$s, it would be better to have a large number of space–time CA-constructible functions at hand.

Acknowledgments

We thank Professor Serge Grigorieff for his indispensable help in revising the manuscript of this paper. We are also indebted to Véronique Terrier who first had the idea of the grouping process. The primitive version of Fig. 5 is due to her.

References

[1] A.J. Atrubin, An iterative one-dimensional real-time multiplier, *IEEE Trans. El. Comp.* **EC-14** (1965) 394–399.
[2] R. Balzer, An 8-state minimal time solution to the Firing Squad Synchronization Problem, *Inform. and Control* **10** (1967) 22–42.
[3] S.N. Cole, Real-time computations by n-dimensional iterative arrays of finite-state machines, *IEEE Trans. Comput.* **18** (1969) 349–365.
[4] K. Culik II and Ch. Choffrut, On real-time cellular automata and treillis automata, *Acta Inform.* **21** (1984) 393–407.
[5] P.C. Fisher, Generation of primes by a real-time iterative array, *J. ACM.* **12** (1965) 388–394.
[6] O.H. Ibarra, S.M. Kim and S. Moran, Sequential machine characterization of treillis and cellular automata and application, *SIAM J. Comput.* **14** (1985) 426–447.
[7] J. Mazoyer, A six state minimal-time solution to the Firing Squad Synchronization Problem, *Theoret. Comput. Sci.* **50** (1987) 183–237.
[8] J. Mazoyer, Variable time FSSP, to appear.
[9] M. Minsky, *Computation: Finite and Infinite Machines* (Prentice Hall, Englewood Cliffs, NJ 1967).
[10] E.F. Moore, *Selected Papers* (Addison-Wesley, Reading, MA, 1964).
[11] A.R. Smith, Real time recognition by one dimensional cellular automata, *J. Comput. System Sci.* **6** (1972) 203–253.
[12] V. Terrier, Décidabilité en Arithmétiques Faibles–Temps Réel sur Automates Cellulaires, Thèse de troisième cycle (Ph.D. dissertation) Ecole Normale Supérieure de Lyon, 1991.
[13] J. Von Neumann, *The theory of self reproducing automata*, edited and complemented by A.W. Burks (University of Illinois Press, Urbana, IL 1967).
[14] A. Waksman, An optimum solution to the Firing Squad Synchronization Problem, *Inform. and Control* **8** (1966) 66–78.