数理解析研究所講究録
754巻 1991年 135-144

On the Power of Two-Dimensional Synchronized Alternating Finite Automata*

コメニウス大学　フロムコビチ　ジュラジュ (Juraj Hromkovic)
山口大学工学部　井上克司 (Katsushi Inoue)
伊藤　暁 (Akira Ito)
高浪五男 (Itsuo Takanami)

ABSTRACT It is well known that four-way two-dimensional alternating finite automata are more powerful than three-way two-dimensional alternating finite automata, which are more powerful than two-way two-dimensional alternating finite automata. This paper shows that four-way, three-way, and two-way two-dimensional "synchronized" alternating finite automata all have the same power as rectangular array bounded automata.

1. Introduction

Synchronized alternation was introduced in [Hr86] as a generalization of the alternation concept from [CKS81] enabling a simple, natural form of communication among parallel processes of alternating devices. Although synchronized alternation is a very new concept, there are already several papers [DHKRS89,HKRS90,IT90,HRS89,HIRST89,Sl88,Sl89,Sl90,Wi89] showing the fruitfulness of this concept.

This paper continues to investigate synchronized alternating devices, especially several types of two-dimensional synchronized alternating finite automata. It is shown in [RS59,Sh59] that "two-way" and "nondeterminism" do not increase the accepting power for (one-dimensional) finite automata, and it is shown in [CKS81,LLS78] that "two-way" and "alternation" also do not increase the accepting power for finite automata. On the other hand, it is shown in [DHKRS89,HKRS90] that "synchronized alternation" drastically increases the accepting power for finite automata, i.e., two-way synchronized alternating finite automata recognize exactly context-sensitive languages. Further, this result has been improved in [HIRST90] by showing that one-way synchronized alternating finite automata do the same.

In this paper, we are mainly concerned with two-dimensional finite automata. The separation results among "determinism", "nondeterminism", and "alternation" for two-dimensional finite automata have been proved in [BH67,ITT83], and the separation results among "four-way", "three-way", and

* This paper was made during the stay of the first author at Yamaguchi University.

"two-way" for two-dimensional finite automata have been proven in [Ro77,IIT88,IIT89]. This paper shows that four-way, three-way, and two-way two-dimensional synchronized alternating finite automata all have the same accepting power as rectangular array bounded automata [IN79]. This result is a natural extension of the results (in [DHKRS89,HKRS90,HIRST90]) described above.

The next section is devoted to the definitions of two-dimensional synchronized finite automata and rectangular array bounded automata. Section 3 gives our main theorem.

2. Definitions

Let Σ be a finite set of symbols. A two-dimensional tape over Σ is a two-dimensional rectangular array of elements of Σ. The set of all two-dimensional tapes over Σ is denoted by $\Sigma^{(2)}$. For each x in $\Sigma^{(2)}$, $\mathcal{Q}_1(x)$ denotes the number of rows of x and $\mathcal{Q}_2(x)$ denotes the number of columns of x.

We refer to [ITT83] for a more formal introduction of a two-dimensional alternating finite automaton (2-AFA). A four-way 2-AFA (FW-2-AFA) M has a read-only (rectangular) input tape with boundary symbols #. Of course, M has a finite control and an input head. A position is assigned to each cell of the read-only input tape, as shown in Fig.1. A step of M consists of reading one symbol from the input tape, moving the input head in direction d ($d \in$ {left,right,up,down,no move}), and entering a new state, in accordance with the next move relation. If the input head falls off the input tape, then M can make no further move. The state set of M is partitioned into accepting, rejecting, existential and universal states.

A four-way two-dimensional synchronized alternating finite automaton (FW-2-SAFA) M is a FW-2-AFA some states of which have a synchronizing element (synchronizing symbol) from some given finite set. These states and the configurations (see below) associated with them are called synchronizing states and synchronizing configurations, respectively. When a process P enters a synchronizing state, it stops and waits until all parallel processes either enter the states with the same synchronizing element or stop in accepting states.

A configuration of a FW-2-SAFA M is of the form (x,(q,(i,j))), where x is the two-dimensional input tape, q is the state of the finite control, and (i,j) is the input head position ($0 \leq i \leq \mathcal{Q}_1(x)+1$, $0 \leq j \leq \mathcal{Q}_2(x)+1$). [If x is clear from the context, the configuration (x,(q,(i,j))) is abbreviated by (q,(i,j)).] The initial configuration of M on input x is $I_M(x)=(x,(q_0,(1,1))$, where q_0 is the initial state of M. A configuration is called existential, universal, accepting, and rejecting, respectively, if the corresponding state is existential, universal, accepting, and reject-

ing, respectively.

Given a FW-2-SAFA M, we write $c \vdash_M c'$ and say c' is a successor of c if the configuration c' follows from the configuration c in one step of M according to the transition rules. A sequence of configurations of M, $c_0, c_1, \ldots, c_m$ $(m \geq 0)$, is called a sequential computation of M if $c_0 \vdash_M c_1 \vdash_M \ldots \vdash_M c_m$. If $c_0 = I_M(x)$ for some x, we call this sequence a computation path of M on x. Let $\bar{c}$ be a sequential computation of M and $c_1, c_2, \ldots, c_r$ be a subsequence of $\bar{c}$ which consists of all synchronizing configurations of $\bar{c}$. For each j $(1 \leq j \leq r)$, let S_j be the synchronizing element of the synchronizing state in c_j. Then, the sequence $S_1, S_2, \ldots, S_r$ is called the synchronizing sequence of $\bar{c}$.

A computation tree of M is a (possibly infinite) labelled tree with the following properties:

(1) Each node v of the tree is labeled with a configuration $\mathcal{Q}(v)$.

(2) If v is an internal node (a non-leaf) of the tree, $\mathcal{Q}(v)$ is universal and $\{c \mid \mathcal{Q}(v) \vdash_M c\} = \{c_1, \ldots, c_k\}$, then v has exactly k children $v_1, \ldots, v_k$ such that $\mathcal{Q}(v_i) = c_i$ $(1 \leq i \leq k)$.

(3) If v is an internal node of the tree and $\mathcal{Q}(v)$ is existential, then v has exactly one child u such that $\mathcal{Q}(v) \vdash_M \mathcal{Q}(u)$.

(4) For any two synchronizing sequences $S = S_1, \ldots, S_p$ and $T = T_1, \ldots, T_r$ corresponding to two paths of the tree beginning at the root, it must be satisfied that $S_i = T_i$ for each $i \in \{1, 2, \ldots, \min\{p, r\}\}$.

A computation tree of M on input x is a computation tree of M whose root is labeled with $I_M(x)$. An accepting computation tree of M on x is a computation tree of M on x whose leaves are all labeled with accepting configurations. We say that M accepts x if there is an accepting computation tree of M on x. Let $T(M) = \{x \mid M \text{ accepts } x\}$.

A three-way two-dimensional synchronized alternating finite automaton (TR-2-SAFA) is a FW-2-SAFA whose input head cannot move up, and a two-way two-dimensional synchronized alternating finite automaton (TW-2-SAFA) is a FW-2-SAFA whose input head can move neither left nor up.

A rectangular array bounded automaton (RABA) is an extension of a linear bounded automaton [HU69] to two dimensions. That is, an RABA M has a finite control, a read-write two-dimensional tape with boundary symbols #, and a tape head (see Fig.2). A position is assigned to each cell of the tape, as shown in Fig.2. Initially, the tape of M includes its input x, and M starts in its initial state with its tape head on the upper-left corner of x. A step of M consists of reading a symbol on the tape cell under the tape head, writing a new symbol on the tape cell, moving the tape head in specified direction (one of left, right, up, down, and no move), and entering a new state, in accordance with the next-move function. M accepts the input x if it eventually enters an accepting state. The set of all two-dimensional tapes accepted by M is denoted by T(M). In general, M is nondeterministic. (See [IN79] for the formal definition of RABA. We note that M cannot rewrite boundary symbols # by another symbols, and the tape head

of M cannot move out of the boundary symbols #.)

For each X∈ {FW-2-SAFA,TR-2-SAFA,TW-2-SAFA,RABA}, let $\mathcal{L}$[X] denote the class of sets accepted by X's. Thus, for example, $\mathcal{L}$[FW-2-SAFA]={T | T=T(M) for some FW-2-SAFA M}.

3. Results

It is shown in [IIT88,IIT89] that four-way two-dimensional alternating finite automata are more powerful than three-way two-dimensional alternating finite automata, which are more powerful than two-way two-dimensional alternating finite automata. In this section, we show that the drastic two-way restriction on the movement of the head of the two-dimensional synchronized alternating finite automaton does not decrease the accepting power. In fact, we show that FW-2-SAFA's, TR-2-SAFA's, and TW-2-SAFA's all have the same accepting power as RABA's.

We first give several notions necessary for our proof. For each synchronized alternating device M, a computation tree of M is defined as in the previous section.

Definition 3.1. Let t be a computation tree of a synchronized alternating device. The synchronization depth of a node v of t is the number of synchronizing configurations on the path from the root to v (excluding the configuration which is the label of v). A meaningful cut of t is a set Z of nodes in t having the same synchronizing depth d such that every infinite path from the root and every path from the root to a leaf node with synchronization depth greater than d contain exactly one node from Z. A synchronization cut of a computation tree t is a meaningful cut containing nodes labelled by synchronizing configurations only.

Before starting to prove our result, we present computing techniques in order to explain our proof in a structured, readable form. For brevity of description, we describe the computing techniques for one-dimensional synchronized alternating devices. The techniques can easily be extended for two-dimensional synchronized alternating devices.

Despite the fact that the definition of synchronization is uniform, i.e., all parallel processes must take part, we can achieve that in fact we synchronize only specific processes with the rest in effect idling. The idea of "idling techniques" is based on adding for each state s of a synchronized alternating machine a special state s' called idling counterpart of s.

Let us first introduce the simpler version of "idling", so called "deterministic idling". This version of idling is called deterministic because each process decides deterministically whether it will be active or idling in the synchronizing period. Suppose that we have three processes A,

B, and C (one may consider a group of processors instead of B or C) and we want A and B to synchronize by some sequence of synchronizing states. Let us assume that both A and B know that they want to synchronize each other, and the process C knows that it has to be idling because the other processes want to synchronize. So, all the processes A, B, and C deterministically produce a special synchronizing symbol S_B (which denotes the beginning of the synchronizing period), and after that A and B are engaged in the synchronization, and C deterministically enters the idling counterpart of its current state. In this idling state, C keeps on guessing the sequence of synchronizing symbols used by A and B (entering synchronizing states with the given idling state and corresponding synchronizing symbols). When the synchronizing period of A and B is over, A and B deterministically produce a special synchronizing symbol S_E (which denotes the end of the synchronization period). C nondeterministically guesses the synchronizing symbol S_E, leaves its idling state, and enters its "active" counterpart.

Now, let us introduce a little more complicated idling technique. We assume that the process A wants to communicate by synchronization with all such processes from the set of all working parallel processes $B_1,\ldots,B_k$ that have their input heads at the same position of the input tape as A has, and all processes $B_1,\ldots B_k$ know it and are prepared to cooperate with A. A starts deterministically by producing a special synchronizing symbol S_B. Each one of the processes $B_1,\ldots,B_k$ deterministically produces S_B, and nondeterministically decides either to remain active or to enter an idling counterpart of its state. Let the processes $C_1,\ldots,C_m \in \{B_1,\ldots,B_k\}$ remain active (i.e., they guess that they have their input heads at the same position as A has) and the processes $D_1,\ldots,D_v \in \{B_1,\ldots,B_k\}$ decide to be idling (i.e., they guess that they have their input heads at another positions as A has). Now, we can assume that all processes $D_1,\ldots,D_v$ will be idling until a special synchronizing symbol S_1 is produced, and the processes $A,C_1,\ldots,C_m$ can start to check whether they have the input heads at the same position. Each $X \in \{A,C_1,\ldots,C_m\}$ universally splits itself into X and X', and X enters the idling counterpart of its state remaining there until the synchronizing symbol S_1 is produced. Each X' starts to move its input head to the right, and produces the synchronizing symbol S_c in each step. When X' reaches the right endmarker \$, it produces the synchronizing symbol S_1 and stops in an accepting state. Since each X' produces so many synchronizing symbols S_c as the distance of the head of X to the right endmarker \$, the computation can continue only if the heads of all processes $A,C_1,\ldots,C_m$ coincide.

After the synchronizing cut labelled by the synchronizing symbol S_1, the computation starts to check whether the head of A does not coincide with any head of processes $D_1,\ldots,D_v$. Now, the processes $C_1,\ldots,C_m$ start again to be idling, and each $Y \in \{A,D_1,\ldots,D_v\}$ universally splits itself into Y and Y. Each Y deterministically enters its idling counterpart. After that,

A moving its input head to the right produces so many synchronizing symbols S_c as the distance, say d, of its head to the right endmarker \$, and A stops in an accepting state after producing a special synchronizing symbol S_2. For each $i \in \{1,\ldots,v\}$, D_i existentially decide if it produces several (at least one) synchronizing symbols S_c without a head movement or it moves its head some (at least one) cells to the right without producing synchronizing symbols. Afterwards D_i starts to work deterministically by moving its input head to the right and by producing the synchronizing symbol S_c in each step. When D_i reaches the right endmarker, it produces the synchronizing symbol S_2 and stops in an accepting state.

Unlike A each D_i produces the greater or less number of the synchronizing symbols S_c (before S_2 is produced) than the distance of its head to the right endmarker. Obviously, if there is such D_i (D_i), for $i \in \{1,\ldots,v\}$, that the head of D_i coincides with the head of A, then this synchronization cannot be successful. On the other hand, if the distance of the head of D_i to the right endmarker is d+r (d-r) for some $r \geq 1$, then D_i nondeterministically moves its head r cells to the right (or produces r symbols S_c, resp.), before it becomes deterministic. In such case D_i produces exactly d synchronizing symbols S_c, and the checking procedure is successful.

After the synchronizing cut labelled by S_2, we have only the original processes $A,C_1,\ldots,C_m,D_1,\ldots,D_v$, and we are already in the same situation as in deterministic idling. Now, the processes $D_1,\ldots,D_v$ will be idling and the processes $A,C_1,\ldots,C_m$ will communicate with each other by synchronization. When the synchronization period of $A,C_1,\ldots,C_m$ is over, then the special synchronizing symbol S_E (end of synchronization) is produced, all processes $A,C_1,\ldots,C_m,D_1,\ldots,D_v$ are active again and can continue in the computation.

Note that the idling technique works also for devices with one-way input tape because we have used only movements to the right in the checking procedures. We turn your attention to the fact that the idling technique can be used also for the following situations:

The process A wants to communicate by synchronization with

1. all processes which are in the same state
2. all processes which have the same contents on the working tape (or on another type of memory)
3. all processes reading the same symbol on the input (working) tape
4. all processes which have the same position of the head on the working tape (the same length of the working tape, pushdown, counter, etc.).

We are now ready to prove our result.

Theorem 3.1. $\mathcal{L}$[TW-2-SAFA]=$\mathcal{L}$[TR-2-SAFA]=$\mathcal{L}$[FW-2-SAFA]=$\mathcal{L}$[RABA].

Proof. The inclusions $\mathcal{L}$[TW-2-SAFA]$\subseteq$ $\mathcal{L}$[TR-2-SAFA]$\subseteq$ $\mathcal{L}$[FW-2-SAFA] are trivial. In order to prove our theorem, we shall show that $\mathcal{L}$[FW-2-SAFA]$\subseteq$

$\mathcal{L}$[RABA] and $\mathcal{L}$[RABA]$\subseteq\mathcal{L}$[TW-2-SAFA].

Let us first show that $\mathcal{L}$[FW-2-SAFA]$\subseteq\mathcal{L}$[RABA]. To do so, we shall use the simulation technique from [DHKRS'89]. Let A be a FW-2-SAFA. It is shown in [DHKRS'89] that each synchronized alternating device B can be simulated by any nondeterministic device which is able to store the set of all distinct configurations of any meaningful cut of B, and to simulate one step from one configuration to its successor. Thus, we first show that an RABA M is able to store any subset S of the set of all configurations of A. To store a configuration (q,(i,j)) of A, M writes the symbol {q} on the (i,j)-cell of its tape. That is, each cell of the tape of M will contain a symbol representing a subset (possibly empty) of the finite set of all states of A. If a cell (i,j) contains a set $\{p_1,\ldots,p_k\}$, it means that the configurations $(p_1,(i,j)),\ldots,(p_k,(i,j))$ are in the given subset S (meaningful cut) of the set of all configurations of A. Using this representation of meaningful cuts of the computations of A, one can easily see that M is able to change (in its representation) any configuration of A by its successor in the computation of A. Thus, we have $\mathcal{L}$[FW-2-SAFA]$\subseteq\mathcal{L}$[RABA].

Now, let us show that $\mathcal{L}$[RABA]$\subseteq\mathcal{L}$[TW-2-SAFA]. Let A be an RABA. We shall construct a TW-2-SAFA B which simulates A. Given an input tape w with $\mathcal{Q}_1(w)=m$ and $\mathcal{Q}_2(w)=n$, B uses one parallel process $B_{i,j}$ for each tape position (i,j) ($i\in\{0,1,\ldots,m+1\}$ and $j\in\{0,1,\ldots,n+1\}$). $B_{i,j}$ keeps its input head stationary on the (i,j)-cell of the input tape through the whole simulation, and $B_{i,j}$ stores the current symbol $b_{i,j}$ on the (i,j)-cell of the tape of RABA A in its finite control. [It will be obvious that any contents of the tape of A can be stored by B in this way.] Furthermore, B uses one parallel process H positioned at the current head position of A and storing the current internal state of A. Another special parallel process P is positioned at the upper-left corner of the input w through the whole simulation, and we use it to simulate the movement of the head of A to the left and up. During the simulation several another auxiliary processes will be used in order to check the coincidence of head positions of H and some $B_{i,j}$. Clearly, the parallel processes (finite automata) H and $B_{i,j}$'s unambiguously represent a configuration of A. So, we can start to describe the computation of B that simulates one step of A from one configuration to its successor. Note that the initial configuration of A on an input tape can be obtained by a gradual splitting of B into $B_{i,j}$'s with each $B_{i,j}$ adjusted exactly at the (i,j)-cell of the input.

1. Let all the processes P, H, and $B_{i,j}$'s be synchronized with a special synchronizing symbol S_1 (S_1 denotes the beginning of the simulation of one step of A). After that, P starts to be idling.

2. Suppose that the head of H is positioned at an (r,s)-cell for $r\in\{0,1,\ldots,m+1\}$ and $s\in\{0,1,\ldots,n+1\}$. All $B_{i,j}$'s except $B_{r,s}$ are supposed to enter their idling states and to remain idling until a special synchronizing symbol S_1' is produced (S_1' denotes the end of the simulation of one step of A). $B_{r,s}$ nondeterministically guesses the current state q of A, H

nondeterministically guesses $b_{r,s}$ (the symbol scanned by $B_{r,s}$ on the (r,s)-cell), and they confirm the guesses by synchronizing themselves by the synchronizing symbol $(q,b_{r,s})$.

3. $B_{r,s}$ and H check the coincidence of their head positions by the idling technique described above. [If another parallel process $B_{u,v}$ ($u\neq r$ or $v\neq s$) decided to be active in step 2, then $B_{u,v}$ is also required to check the coincidence of its head position with the head position of H. Clearly, the head positions do not coincide, and the computation cannot be accepting in this case.] Let the end of the checking of the head coincidence be marked by a special synchronizing symbol S_2. After that, only H and $B_{r,s}$ are active.

4. Now, both H and $B_{r,s}$ know the current state of A and the symbol read by A. Since A is nondeterministic, H and $B_{r,s}$ nondeterministically guess the same action (p,b,Z) (where p is a new state, b is a new symbol on the (r,s)-cell, and $Z\in$ {left,right,up,down,no move} describes the next movement of the head of A), and check this common guessing by producing [p,b,Z] as a synchronizing symbol. After that, P starts to be active, P and H remember the new state p, and $B_{r,s}$ stores the symbol b instead of $b_{r,s}$ in its finite control.

5. If $Z\in$ {right,down,no move}, then H simulates the movement of the head of A and the simulation continues in the following step 6. If $Z\in$ {left,up} (i.e., the head moves to the left or up), then $B_{r,s}$ starts to be idling, and P splits itself into two copies P and P'. Then P starts to be idling and P' starts to move its head to the right and down until P' nondeterministically guesses that its head is one cell left from the head of H (if Z=left) or that its head is one cell up from the head of H (if Z=up). Using the idling technique, the correctness of the guess of P' is checked, and H finishes its role in an accepting state. Now, P' starts to play the role of H in the next simulation step.

6. All processes P, H (P'), $B_{i,j}$'s produce the synchronizing symbol S_1' (which denotes the end of the simulation of one step of A), and all these (m+2)(n+2)+2 processes again become active in order to simulate the next step of A.

If the state p (see step 4) is an accepting state of A, then the processes H and $B_{r,s}$ produce a special synchronizing symbol S_E (which denotes the end of the computation). After that, all parallel processes finish their works in an accepting state.

Obviously, A accepts an input w iff B does it. This completes the proof of "$\mathcal{L}$[RABA]$\subseteq\mathcal{L}$[TW-2-SAFA]". Q.E.D.

REFERENCES

[BH67] M.Blum and C.Hewitt, Automata on a two-dimensional tape, IEEE Symposium on Switching and Automata Theory, 1967, 155-160.
[CKS81] A.K.Chandra,D.K.Kozen, and J.Stockmeyer, Alternation, J.ACM 28(1)(1981), 114-133.
[DHKRS89] J.Dassow,J.Hromkovic,J.Karhumaki,B.Rovan, and A.Slobodova, On the power of synchronization in parallel computations, Proc.14th MFCS'89, Lecture Notes in Computer Science 379, Springer-Verlag 1989, 196-206.
[Hr86] J.Hromkovic, How to organize the communication among parallel processes in alternation computations, Unpublished manuscript, Comenius University, Bratislava, January 1986.
[HKRS89] J.Hromkovic,J.Karhumaki,B.Rovan, and A.Slobodova, On the power of synchronization in parallel computations, to appear in Discrete Applied Mathematics.
[HIRSTW90] J.Hromkovic,K.Inoue,B.Rovan,A.Slobodova,I.Takanami, and K.R.Wagner, On the power of one-way synchronized alternating machines with small space, Submitted to International Journal of Foundations of Computer Science.
[HRS89] J.Hromkovic,B.Rovan, and A.Slobodova, Deterministic versus nondeterministic space in terms of synchronized alternating machines, Submitted to Mathematical Computation Theory.
[HU69] J.E.Hopcroft and J.D.Ullman, Formal Languages and their Relation to Automata, Addison-Wesley, Reading Mass.(1969).
[IIT88] A.Ito,K.Inoue, and I.Takanami, A note on three-way two-dimensional alternating Turing machines, Information Sciences 45(1)(1988), 1-22.
[IIT89] A.Ito,K.Inoue,and I.Takanami, Deterministic on-line tessellation acceptors are equivalent to two-way two-dimensional alternating finite automata through 180 rotations, Theoretical Computer Science 66(1989), 273-287.
[IN79] K.Inoue and A.Nakamura, Two-dimensiponal multipass on-line tessellation acceptors, Information and Control 41(3)(1979), 305-323.
[IT90] O.H.Ibarra and N.Q.Tran, Unpublished manuscript (1990).
[ITT83] K.Inoue, I.Takanami, and H.Taniguchi, Two-dimensional alternating Turing machines, Theoretical Computer Science 27 (1983), 61-83.
[LLS78] R.E.Ladner,R.J.Lipton, and L.J.Stockmeyer, Alternating pushdown automata, Proc.19th IEEE Symp. on Foundations of Computer Science, Ann Arbor, MI (1978).
[Ro77] A.Rosenfeld, Picture Languages (Formal Models for Picture Recognition), Academic Press, New York (1977).
[RS59] M.O.Rabin and D.Scott,Finite automata and their decision problems, IBM.J.Res.3:2 (1959), 115-125.
[Sh59] J.C.Shepherdson, The reduction of two-way automata to one-way automata, IBM.J.Res.3 (1959), 198-200.
[Sl88] A.Slobodova, On the power of communication in alternating machines, Proc.13th MFCS's, Lecture Notes in Computer Science 324, Springer-Verlag 1988, 518-528.

[Sl89] A.Slobodova, Some properties of space-bounded synchronized alternating Turing machines with only universal states, In: Machines,Languages and Complexity (J.Dassow,J.Kelemen(Eds.)), Lecture Notes in Computer Science 381, Springer-Verlag 1989, 102-113.
[Sl90] A.Slobodova, One-way globally deterministic synchronized alternating finite automata recognize exactly deterministic context-sensitive languages, Information Processing Letters 36 (1990), 69-72.
[Wi89] J.Wiedermann, On the power of synchronization, J.Inf.Process.Cybern.EIK 25(10)(1989), 499-506.

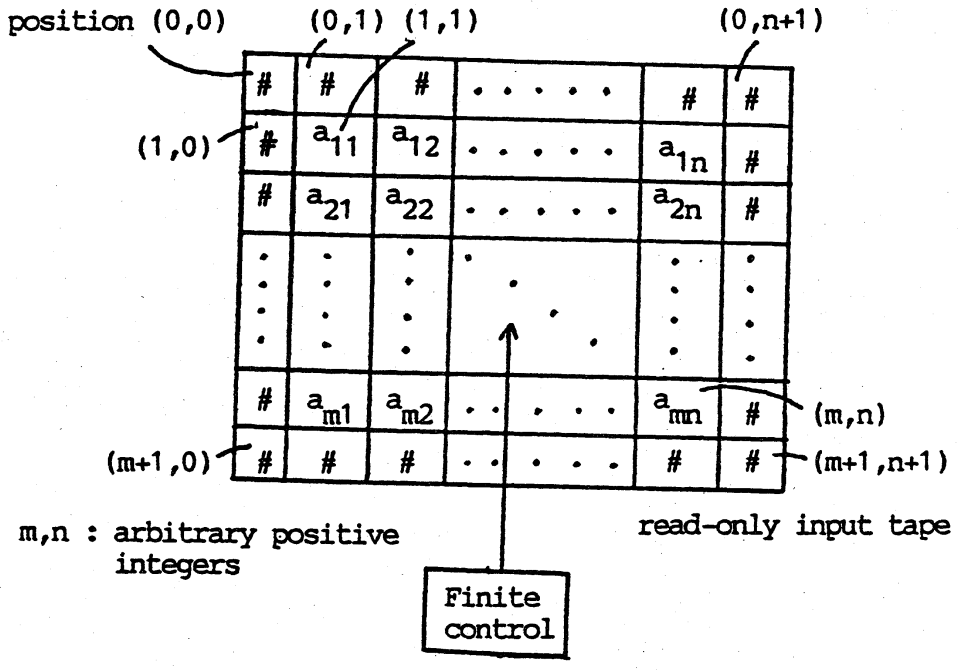

Fig.1. Two-dimensional alternating finite automaton