FINITE AUTOMATA WITH MARKERS

by

Pei Hsia and Raymond T. Yeh

March 1973 TR-12

FINITE AUTOMATA WITH MARKERS*

by

Pei Hsia
Logican, Inc.
Los Angeles, Calif.

and

Raymond T. Yeh
Dept. of Computer Sciences
The University of Texas at Austin
Austin, Texas 78759

*This paper is supported in part by the National Science Foundation grant GJ-31528 and by the Joint Services Electronics Program under Research Grant F44620-71-C-0091. "This paper has been presented at the International Symposium on Theory of Automata and Programming Language in France, 1972."

I. Introduction

This paper studies finite automata augmented with markers which the automata can move about on their input tapes. The concept of augmenting markers to automata was first introduced by Blum and Hewitt [1] in two-dimensional automata. Kreider, Ritchie and Springsteel [6,7,8,12] investigated recognition of context-free languages by (one-dimensional) automata with markers. In this paper, we investigate some fundamental properties of marker automata and study their relationships to other types of automata and languages.

The main result in this paper is the establishment of an infinite hierarchy of languages recognizable by deterministic and deterministic, halting marker automata. It also turns out that because of the equivalence of finite marker automata and multi-head automata, the study of three-marker automata becomes very interesting due to results of Hartman's [4] and Savitch [10].

II. General Properties of Marker Automata.

In this section we will prove the equivalence of labelled and unlabelled marker automata and show that there is an infinite hierarchy in the class of languages acceptable by deterministic marker automaton. Furthermore, it will be whown here that every deterministic k-marker automaton is equivalent to a deterministic, halting (2k+3)-marker automaton.

Definition 1. Let $\underline{N}$ denote the set of natural numbers and $K = \{1,2,...k\}$. A deterministic (non-deterministic) k-marker automaton A is a 7-tuple $[S,\Sigma,K,\delta$ $s_o,s_a,s_r]$ where S and Σ are finite sets of states and input symbols, respectively. Symbols s_o,s_a,s_r, are elements of S called the initial state, accepting state, and rejecting state, respectively, and δ is a function mapping $S \times \{\Sigma \cup K\}$ to (finite subsets of) $S \times \{\{-1\}\cup\{0\}\cup K\} \times \{-1,0,1\}$. A is referred to as an ∞-marker automaton if K is replaced by $\underline{N}$.

Intuitively, the function of δ is to change state, remove from or place a marker on the input tape, and move the reading head. Note that if a number of markers are stacked up on an input square, the reading head of the automaton will only read the marker on top of the stack. However, the automaton can sense when more than one marker are stacked on a particular square. The input alphabet Σ also contains a special symbol ¢ used as end marker of input tapes to keep the reading head from falling off. The automaton stops whenever it enters state s_a or s_r. If A is an ∞-marker automaton, then the number of markers to be stacked on a square can be arbitrarily large.

Definition 2. The language acceptable by a k-marker automaton $A = [S,\Sigma,$ $M,\delta,s_o,s_a,s_r]$ consists of all tapes x over Σ-¢ such that starting with state s_o on the left most symbol of x, A eventually halts on state s_a.

Definition 3. A deterministic (non-deterministic) unlabelled k-marker automaton B is a 6-tuple $[S,\Sigma,\lambda,s_o,s_a,s_r]$, where S,Σ,s_o,s_a,s_r are as defined in definition 1, and λ is a function mapping $S \times \{\Sigma \cup \{*\}\}$ into (finite subsets of) $S \times \{-1,0,1\} \times \{-1,0,1\}$. B is called an unlabelled ∞-marker automaton if it has an infinite number of unlabelled markers.

We note that the symbol * in definition 3 indicates the presence of a marker.

Definition 4. Two (marker) automata are said to be equivalent if they accept the same language.

Theorem 1. Automata with unlabelled markers are equivalent to automata with labelled markers.

Proof: Clearly an automaton with labelled markers can simulate an automaton with the same number of unlabelled markers.

Consider now a labelled k-marker automaton $A = [S,\Sigma,K,\delta,s_o,s_a,s_r]$. Construct an unlabelled k-marker automaton $B = [S',\Sigma,\lambda,s_o',s_a',s_r']$ to simulate A such that each state of B contains the following information about A: 1) Relative positions of markers of A on input tapes and in stacks; 2) Relative positions of markers of A and reading head of A; 3) states of A. More specifically, let M_1 denote the set of sequences of distincet elements of K of length less than or equal to k, and $M_2 = \{\alpha_1\ldots\alpha_i\text{-}\beta\text{-}\gamma_1\ldots\gamma_j \mid \alpha_1\ldots\alpha_i\beta\gamma_1\ldots\gamma_j \in M_1\}$. Then we define B as follows:

$S' = \{(s^*,\alpha) \mid s^* \in \{s,\vec{s},\overleftarrow{s} \mid s \in S\}$ and $\alpha \in M_2\}$.

$s_o' = (s_o,\text{--})$, and λ is defined so that $\lambda((s,\alpha_1,\ldots\alpha_i\text{-}\beta\text{-}\gamma_1\ldots\gamma_j),a)$ contains

Case 1. $\beta = \Lambda$ and $a \in \Sigma$

$((t,\alpha_1\ldots\alpha_i\text{--}\gamma_1\ldots\gamma_j),0,0)$, if $(t,0,0) \in \delta(s,a)$

$((\vec{t},\alpha_1\ldots\alpha_i--\gamma_1\ldots\gamma_j),0,1)$, if $(t,0,1) \in \delta(s,a)$

$((\overleftarrow{t},\alpha_1\ldots\alpha_i--\gamma_1\ldots\gamma_j),0,-1)$, if $(t,0,-1) \in \delta(s,a)$

$((t,\alpha_1\ldots\alpha_i--p\gamma_1\ldots\gamma_j),1,0)$, if $(t,p,0) \in \delta(s,a)$

$((\overleftarrow{t},\alpha_1\ldots\alpha_i--p\gamma_1\ldots\gamma_j),1,-1)$, if $(t,p,-1) \in \delta(s,a)$

$((\vec{t},\alpha_1\ldots\alpha_i p--\gamma_1\ldots\gamma_j),1,1)$, if $(t,p,1) \in \delta(s,a)$

Case 2. $\beta = P_1\ldots P_{n-1}P_n = \beta' P_n, a = *$

$((t,\alpha_1\ldots\alpha_i-\beta-\gamma_1\ldots\gamma_j),0,0)$, if $(t,0,0) \in \delta(s,P_n)$

$((\vec{t},\alpha_1\ldots\alpha_i\beta--\gamma_1\ldots\gamma_j),0,1)$, if $(t,0,1) \in \delta(s,P_n)$

$((\overleftarrow{t},\alpha_1\ldots\alpha_i--\beta\gamma_1\ldots\gamma_j),0,-1)$, if $(t,0,-1) \in \delta(s,P_n)$

$((t,\alpha_1\ldots\alpha_i--\beta P-\gamma_1,\ldots\gamma_j),1,0)$, if $(t,P,0) \in \delta(s,P_n)$

$((\vec{t},\alpha_1\ldots\alpha_i(\beta P)--\gamma_1,\ldots\gamma_j),1,1)$, if $(t,P,1) \in \delta(s,P_n)$

$((\overleftarrow{t},\alpha_1\ldots\alpha_i--(\beta P)\gamma_1\ldots\gamma_j),1,-1)$, if $(t,P,-1) \in \delta(s,P_n)$

$((t,\alpha_1\ldots\alpha_i-\beta'-\gamma_1\ldots\gamma_j),-1,0)$, if $(t,-1,0) \in \delta(s,P_n)$

$((\vec{t},\alpha_1\ldots\alpha_i\beta'--\gamma_1\ldots\gamma_j),-1,1)$if $(t,-1,1) \in \delta(s,P_n)$

$((\overleftarrow{t},\alpha_1\ldots\alpha_i--\beta'\gamma_1\ldots\gamma_j),-1,-1)$, if $(t,-1,-1) \in \delta(s,P_n)$

Case 3. for $a \in \Sigma$ and $s^* \in \{\vec{s},\overleftarrow{s},\}$

$\lambda((s^*,\alpha_1\ldots\alpha_i--\gamma_1\ldots\gamma_j),a) = \lambda((s,\alpha_1\ldots\alpha_i--\gamma_1\ldots\gamma_j),a)$

Case 4. $\lambda((\overleftarrow{s},\alpha_1\ldots\alpha_i--\gamma_1\ldots\gamma_j),*) = \lambda((s,\alpha_1\ldots\alpha_{i-1}-\alpha_i-\gamma_1\ldots\gamma_j),*)$

Case 5. $\lambda((\vec{s},\alpha_1\ldots\alpha_i--\gamma_1\ldots\gamma_j),*) = \lambda((s,\alpha_1\ldots\alpha_i-\gamma_1-\gamma_2\ldots\gamma_j),*)$

Finally, B enters s_a or s_r whenever A does.

It is easily seen that the constructed automaton B does simulate A.

In the case A is an ∞-marker automaton whose markers are elements of $\underline{N}$, we construct an unlabelled ∞-marker automaton B to simulate A by assigning a stack of P (unlabelled) markers to simulate a marker P in A. The construction of B is simple but lengthy and hence is omitted here.‖

In the sequel, we will use freely the notion of marker automaton without mentioning whether the markers are labelled or not. (It should be clear from the context as to what types of markers are used.)

The following two theorems are known results [1] which are included here for the sake of completeness.

Theorem 2. 0-marker automata are equivalent to 1-marker automata.

Theorem 3. Automata with k-markers which can place at most one marker on any square of input tape is as powerful as k-marker automata which can stack markers on a single square.

We would like to remark here that it is still an open problem whether nondeterministic marker automaton is more powerful than a deterministic automaton having the same number of markers. The answer is affirmative in the case of two-dimensional marker automaton [1].

In the following, we will prove two important properties of deterministic marker automata.

Definition 4. A language L is said to be n-marker acceptable if L = L(A) for some n-marker automaton A. It is called an n-marker deterministic language in case A is deterministic. It is called an n-marker deterministic, halting language in case A also halts on all inputs.

In the following, a sequence of lemmas will be proved in order to establish the main result of an infinite hierarchy in the family of deterministic, halting languages.

Let L be an n-marker dterminiistic, halting language over Σ such that L is not (n-i)-marker acceptable for any $i>0$. Construct two languages L' and L" as follows:

$L' = \{\alpha_1\#\alpha_2\#\ldots\#\alpha_m | m>0,\ \alpha_i \in \Sigma^*,\ \# \notin \Sigma$, and number of α_i's in L is exactly the same as number of α_j's not in L for $1 \leq i,\ j \leq m.\}$ (1)

$L'' = \{\alpha\#x | \alpha \in \Sigma^*$ and x is a fixed string not in L.$\}$ (2)

Lemma 1. If A is a deterministic, halting k-marker automaton such that $L(A) = L''$ then A can decide if $\alpha \in \Sigma^*$ is in L or not.

Proof: Indeed, $\alpha \in L$ if an only if $\alpha\#x \in L''$; where x is the fixed string given in (2). ||

Lemma 2. Let A be an n-marker automaton. If $L(A) = L'$, then there exists a string $\beta = \alpha_1\#\alpha_2\#\ldots\#\alpha_m \in L'$ such that in the computation of A on β, there is a computational step in which A places all n markers on α_i for some $1 \leq i \leq m$.

Proof: Suppose the statement of the lemma is false. Then, restricting our inputs to L", we may assume that $A = [S,\Sigma,\delta,s_o,s_a,s_r]$ uses at most (n-1) markers on the substring α. Since x is a fixed string, the total number of ways of placing at most n markers on x with distinct states of A is finite. Using this information of state-marker configurations, we can construct an (n-1)-marker automaton B such that the states of B keep track of the state-marker configurations of A on the substring x and uses only its n-1 markers to simulate the movements of markers on the substring α. More specifically,

$$B = [S',\Sigma,\gamma,\overleftarrow{s}_o\#\overbrace{00\ldots0}^{\text{P times}},s_a,s_r], \text{ where}$$

$$S' = \{i_1\ldots i_j\bar{s}\ i_{j+1}\ldots i_p | 0 \leq i_j \leq n \text{ and length } (i_1\ldots i_p) = n\} \cup$$

$$\{\bar{s}\#i_1\ldots i_p | 0 \leq i_j \leq n \text{ and length } (i_1\ldots i_p) = n\} \cup \{s_a,s_r\}$$

where $\bar{s} \in \{\vec{s},\overleftarrow{s} | s \in S\}$ and P is the length of x.

$L' = \{\alpha_1 \# \alpha_2 \# \ldots \# \alpha_m | m>0,\ \alpha_i \in \Sigma^*,\ \# \notin \Sigma$, and number of α_i's in L is exactly the same as number of α_j's not in L for $1 \leq i,\ j \leq m.\}$ (1)

$L'' = \{\alpha \# x | \alpha \in \Sigma^*$ and x is a fixed string not in L.$\}$ (2)

Lemma 1. If A is a deterministic, halting k-marker automaton such that $L(A) = L''$ then A can decide if $\alpha \in \Sigma^*$ is in L or not.

Proof: Indeed, $\alpha \in L$ if an only if $\alpha \# x \in L''$; where x is the fixed string given in (2). ||

Lemma 2. Let A be an n-marker automaton. If $L(A) = L'$, then there exists a string $\beta = \alpha_1 \# \alpha_2 \# \ldots \# \alpha_m \in L'$ such that in the computation of A on β, there is a computational step in which A places all n markers on α_i for some $1 \leq i \leq m$.

Proof: Suppose the statement of the lemma is false. Then, restricting our inputs to L", we may assume that $A = [S,\Sigma,\delta,s_o,s_a,s_r]$ uses at most (n-1) markers on the substring α. Since x is a fixed string, the total number of ways of placing at most n markers on x with distinct states of A is finite. Using this information of state-marker configurations, we can construct an (n-1)-marker automaton B such that the states of B keep track of the state-marker configurations of A on the substring x and uses only its n-1 markers to simulate the movements of markers on the substring α. More specifically,

$$B = [S',\Sigma,\gamma,\overleftarrow{s}_o \# \overbrace{00\ldots0}^{\text{P times}},s_a,s_r], \text{ where}$$

$$S' = \{i_1 \ldots i_j \bar{s}\ i_{j+1} \ldots i_p | 0 \leq i_j \leq n \text{ and length } (i_1 \ldots i_p) = n\} \cup$$

$$\{\bar{s} \# i_1 \ldots i_p | 0 \leq i_j \leq n \text{ and length } (i_1 \ldots i_p) = n\} \cup \{s_a,s_r\}$$

where $\bar{s} \in \{\vec{s},\overleftarrow{s} | s \in S\}$ and P is the length of x.

The transition function γ of B is defined as follows:

$$\gamma(i_1 \ldots i_j \vec{s} i_{j+1} \ldots i_p, a) = \begin{cases} (i_1 \ldots (i_j+d) i_{j+1} \vec{t} i_{j+2} \ldots i_p, d, 1), \text{if } (t,d,1) \varepsilon \delta(s,a) \\ (i_1 \ldots i_{j-1} \vec{t} (i_j+d) \ldots i_p, d, -1), \text{if } (t,d,-1) \varepsilon \delta(s,a) \\ (i_1 \ldots (i_j+d) \vec{t} i_{j+1} \ldots i_p, d, o), \text{if } (t,d,0) \varepsilon \delta(s,a) \end{cases}$$

$$\gamma(\vec{s} i_1 \ldots i_p, \#) = (\overleftarrow{s} \# (i_1+d) \ldots i_p, d, -1), \text{if } (t,d,-1) \varepsilon \delta(s,\#)$$

$$\gamma(\overleftarrow{s} \# i_1 \ldots i_p, a) = \begin{cases} (\overleftarrow{t} \# i_1 \ldots i_p, d_1, d_2), \text{if } (t,d_1,d_2) \varepsilon \delta(s,a), \text{ and} \\ \qquad a \ \varepsilon \ \Sigma \\ (\vec{t} i_1 \ldots i_p, d, 1), \text{if } (t,d,1) \varepsilon \delta(s,\#). \end{cases}$$

Finally, B goes in s_a or s_r whenever it is going to enter a state s' in which $\vec{s}_a$ or $\vec{s}_r$ occurs.

It is easily seen that $L = L(B_{n-1})$. This is a contradiction because L is not (n-1) marker acceptable by assumption. Hence, the lemma is proved. ||

Lemma 3. Let A be a k-marker automaton such that $L(A) = L'$. If $\alpha_1 \# \alpha_2 \# \ldots \# \alpha_{2n} \ \varepsilon \ L'$, then A can decide whether $\alpha_i \ \varepsilon \ L$ for each $1 \leq i \leq 2n$.

Proof: for K=1, the lemma holds by virtue of lemma 1. Assume lemma holds for k>1. Since $\alpha_1 \# \alpha_2 \# \ldots \# \alpha_{2n} \ \varepsilon \ L'$ if and only if $\alpha_{\pi(1)} \# \ldots \# \alpha_{\pi(2)} \# \ldots \# \alpha_{\pi(2n)} \ \varepsilon \ L'$ for any permutation π on $(1,2,\ldots,2n)$, we see that $\alpha_1 \# \alpha_2 \# \ldots \# \alpha_{2n} \ \varepsilon \ L$ implies that $\alpha_{\pi(1)} \# \ldots \# \alpha_{\pi(2n-2)} \ \varepsilon \ L'$ and $\alpha_{\pi(2n-1)} \alpha_{\pi(2n)} \ \varepsilon \ L$ for some permutation π. Since the lemma holds on the substring $\alpha_{\pi(1)} \# \ldots \# \alpha_{\pi(2n-2)}$ by inductive hypothesis, the lemma holds for the whole string using the base of induction.||

Theorem 1. Let L be an n-marker $(n \geq 1)$ deterministic, halting language, then the language L' defined in (1) is a deterministic, halting (n+1)-marker language but is not an n-marker language.

Proof: Let $L = L(A_n)$ for some deterministic, halting n-marker automaton

A_n. We can construct a deterministic, halting (n+1)-marker automaton A_{n+1} such that $L(A_{n+1}) = L'$. The function of A_{n+1} on an input $\alpha_1 \# \alpha_2 \# \ldots \# \alpha_m$ is described in the following:

(a) A_{n+1} uses n markers to check if each $\alpha_i \in L$, $1 \leq i \leq m$.

(b) A_{n+1} uses the remaining marker to keep track of the number of α_i's that are in L by moving it along the separation symbol # from left to right.

(c) After A_{n+1} has checked each α_i, it uses one marker together with the marker used in (b) to determine whether number of α_i in L is the same as that of those not in L. Since it is easily seen that A_{n+1} can be constructed from A_n, we will omit the definition of A_{n+1} here.

It is easily seen that $L(A_{n+1}) = L'$. We now need to show that L' is not n-marker acceptable.

Suppose now B_n is an n-marker automaton such that $L(B_n) = L'$. By lemma 4, we may assume that B_n processes each input string of the type $\beta = \alpha_1 \# \alpha_2 \# \ldots \# \alpha_m$ by first processing each substring α_i according to their order of appearance. Denote by $D(\alpha_j)$ the number of A_i's in L minus the number of α_i's not in L for $i \leq j$. It is clear that $\beta \in L'$ if and only if $D(\alpha_m) = 0$. Let $\gamma = \alpha_1 \# \alpha_2 \# \ldots \# \alpha_k$ be an element of L' such that B_n, in processing γ, must use all of its n-markers in α_i. (Such an γ exists by lemma 3.) This means that in processing γ, B_n has only finite capacity in keeping track of $D(\alpha_i)$. However, there is an infinite number of elements in L' satisfying the property stated in lemma 2. Hence, B_n cannot keep track of this number for all elements of L' and therefore cannot accept L'.||

<u>Corollary 1</u>. There is an infinite hierarchy in the family of deterministic, halting marker languages.

<u>Definition 5</u>. A <u>bounded k-counting automaton</u> (BCA-k) is a quintuple

$B = [S, \Sigma, s_o, f, F]$, where S and Σ are finite sets of states and input symbols, respectively. Σ contains a special end marker ¢. $F \subseteq S$ and $s_o \in S$ are the respective final state set and initial state of B, and f is a partial function from $\Sigma \times S \times \{0,+\}^k$ into $S \times \{1,0,-1\}^k \times \{1,0,-1\}$.

Intuitively, we think of B as having k counters $C_1,\ldots,C_k$, each of which can be tested for being zero or positive, and which can be increased or decreased by one as long as the respective counts of each C_i all stay within a lower bound of zero and an upper bound equal to the length of the current input tape. It was shown by Ritchie and Springsteel [8] that every k-marker automaton is equivalent to a bounded k-counting automaton, and that every bounded k-counting automaton is equivalent to a (k+1) -marker automaton.

Theorem 2. Every deterministic k-marker automaton is equivalent to a deterministic, halting (2k+3) -marker automaton.

Proof: Let A be a given deterministic (but not necessarily halting) k-marker automaton with m states. It is easily seen that A can have at most $\sum_{j=0}^{k} C_j^n \cdot n \cdot m$ distinct computational steps on any input tape of length n. Since $\sum_{j=0}^{k} C_j^n \leq n^{k+1}$ and since any BCA-k with P states can count up to Pn^{k+1} on any input tape of length n, it follows from the equivalence of bounded counting automata and marker automata that we can construct a deterministic, halting (2k+3) -marker automaton B with $P(P \geq m)$ states to simulate A as follows:

1. B uses k-markers to mimic the k-markers of A;

2. B uses a special marker used to keep track of the reading head of A;

3. B uses k+2 markers to simulate the behavior of bounded (k+1) -counting automaton such that B halts whenever the simulated count reaches mn^{k+1} on any input tape of length n. The formal definition of B, being quite simple but lengthy, is omitted here.||

Theorem 3. There is an infinite hierarchy in the family of deterministic marker languages.

Proof: Let L_D^k $(L_{D,H}^k)$ denote languages recognizable by deterministic (as halting) k-marker automata. We have $L_D^k \subseteq L_{D,H}^{2k+3} \subsetneq L_{D,H}^{2k+4} \subseteq L_D^{2k+4}$. ||

Note that this hierarchy theorem is exactly the same as the Blum-Hewitt result for the two-dimensional case [1]. However, the proof of our result is constructive whereas Blum and Hewitt gave an existence proof.

III. Marker Automata and Other Types of Automata

We have mentioned in the previous section the result of Ritche and Springsteel [8] that the class of marker automata is the same as the class of bounded counting automata. We will show in this section the relationship between marker automata and multi-head automata, Turing machines, and linear bounded automata.

Definition 6. A two-way k-head automaton is a seven-tuple $A = [S, \Sigma, \delta, f, s_o, s_a, s_r]$, where S and Σ are finite sets of states and input symbols, respectively s_o, s_a, s_r are initial, accepting, and rejecting states respectively. $f\colon S \to K$ is called the head selecting function and $\delta\colon S \times K \times \Sigma \to S \times K \times \{-1,0,1\}$ is the transition function defined as follows:

$$\delta(s, i, a) = (s', j, d) \text{ if } f(s) = i \text{ and } f(s') = j.$$

Intuitively, $\delta(s, i, a) = (s', j, d)$ means that if the automaton is currently in state s and reading the symbol a using its ith head where $f(s) = i$, then it changes state to s' and moves its jth reading head according to d. Again, we assume that input tapes of M are bounded by end markers and that A accepts or rejects a tape depending on whether it enters s_a or s_r.

Lemma 4. Every two-way k-head automaton is equivalent to a deterministic k-marker automaton.

Proof: Let $A = [S, \Sigma, \delta, f, s_o, s_a, s_r]$ be a given two-way k-head automaton. It is seen intuitively that one can use a k-marker automaton B to simulate A by using one marker to keep track of a reading head of A. However, B would have to contain more states since to simulate the situation in which several heads of A are positioned on one square, it is necessary for B to stack up the corresponding markers. The construction of B is very similar to the construction of an unlabeled marker automaton to simulate a labelled marker automaton described in the proof of theorem 1. Hence, we will not give a formal definition of B here since it is too lengthy to define the transition function γ formally. Rather, we will merely describe how B works.

First of all, let M_1 denote the set of sequences of distinct elements of $K = \{1,2,\ldots,k\}$ (of length less than or equal to k), and let $M_2 = \{\alpha_1\ldots\alpha_i - \beta - \gamma_1\ldots\gamma_j \in M_1\}$. The state set S of B contains s_a and s_r and elements of the form $(\bar{s}, i, \alpha)$, $\bar{s} \in \{s, \vec{s}, \overleftarrow{s} \mid s \in S\}$, $i \in \{-1,0,1\}$, $\alpha \in M_2$. Thus, each state of B keeps the following items of information:

a. current state of A by $\bar{s}$;

b. relative positions of reading head of A by α;

c. the next move of the newly designated reading head of A by i; and

d. whether the marker simulating the next designated reading head of A is located to the left, right, or in the square being scanned. These are denoted by the left arrow, right arrow, or no arrow on top of the state symbol of S.

To begin with, B images all of its markers, except the ith one, are placed on the left-hand end marker ¢, where $i = f(s_o)$. Thus, the initial state of B is

$$s_o' = (s_o, 0, 12 \ldots (i-1)(i+1) \ldots k\text{--}).$$

Now, if B is in $s' = (\bar{s}, i, \alpha_1 \ldots \alpha_i - \beta - \gamma_1 \ldots \gamma_j)$ simulating A in state s and the $\delta(s,a) = (t,d)$, where $f(t) = P$. Then B first decides whether P occurs in $\alpha_1\ldots\alpha_i$ or in $\gamma_1\ldots\gamma_j$, shift to appropriate state $((\overleftarrow{t}, i, \alpha)$, (t, i, α) or $(\vec{t}, i, \alpha))$ and search for the stack ξ in which P occurs. After B has found the stack ξ, it removes each marker in ξ, reads the input symbol on this square, and stacks ξ (minus the marker P) back up again. B then moves the marker P according to the direction d and changes state, which reflects the new marker configuration. B is now ready to repeat the same process. ||

Although we do not know whether the converse of the previous lemma holds, it is easy to see that every deterministic k-marker automaton A is equivalent to a two-way (k+1) -head automaton B. This observation, along with lemma 4, yields the following result.

<u>Theorem 4</u>. The class of languages acceptable by two-way multiple head automata is the same as the class of languages acceptable by marker automata with finite number of markers.

In the following, we will show the equivalence of Turing machines and linear bounded automata to classes of ∞-marker automata.

<u>Theorem 5</u>. Every Turing machine is equivalent to an ∞-marker automaton and vice versa.

Proof: We will use the well-known fact that every Turing machine is equivalent to a two-counter machine and show that any two-counter machine can be simulated by an ∞-marker automaton. Recall that a two-counter machine is a Turing machine restricted to its input tape, but with two infinite storage tapes. We assume that input tapes of two-counter machines are bounded by end markers ¢ on the left and \$ on the right. An ∞-marker automaton can simulate the actions of these two counters by stacking up markers on the two end markers at each end of the input tape. More specifically, let $C = [S, \Sigma, \delta, s_o, F]$ be a two-counter machine with $\delta: S \times \Sigma \rightarrow 2^{S \times D \times D \times D}$, where $D = \{-1,0,1\}$ and the first two D's indicate the movement of the reading heads on the storage tapes and the last D denotes the movement of reading head on the input tape. We construct an ∞-marker autonmaton $A = [S', \Sigma, \gamma, [s_o, -, -, -], s_a, s_r]$ as follows.

A contains three classes of markers, one *, two sets containing an infinite number of $\square$ and #. The state set S' and transition function γ are given in the following:

$$S' = \{s_a, s_r\} \cup S \times D' \times D' \times D' \quad \text{where } D' = D \cup \{ - \}.$$

$$\text{a.}\quad \gamma([s,-,-,-],a) = \begin{cases} s_a, \text{ if } (t,d_1,d_2,d_3) \varepsilon\ \delta(s,a) \text{ for some} \\ \qquad t\ \varepsilon\ F. \\ ([t,d_1,d_2,d_3],*,-1), \text{ if } (t,d_1,d_2,d_3) \\ \qquad \varepsilon\ \delta(s,a)\quad a\ \varepsilon\ \Sigma \text{ and } t \notin F \end{cases}$$

$$\text{b.}\quad \gamma([s,d_1,d_2,d_3],a) = \begin{cases} ([s,d_1,d_2,d_3],0,-1), \text{ for } a\ \varepsilon\ \Sigma \\ ([s,-,d_2,d_3],\Box,1), \text{ for } a\ \varepsilon\ \{\Box,\text{¢}\} \\ \qquad \text{and } d_1 = 1 \\ ([s,-,d_2,d_3],-1,1), \text{ if } a = \Box, \text{ and } d_1 = -1 \\ ([s,-,d_2,d_3],0,1), \text{ if } a\ \varepsilon\ \{\Box,\text{¢}\} \text{ and} \\ \qquad d_1 = 0 \\ s_r' \text{ if } a = \text{¢} \text{ and } d_1 = -1 \end{cases}$$

$$\text{c.}\quad \gamma([s,-,d_2,d_3],a) = \begin{cases} ([s,-,d_2,d_3],0,1), \text{ for } a\ \varepsilon\ \Sigma \cup \{*\} \\ ([s,-,-,d_3],\#,-1), \text{ for } a\ \varepsilon\ \{\$,\#\} \text{ and} \\ \qquad d_2 = 1 \\ ([s,-,-,d_3],-1,-1), \text{ for } a = \# \text{ and } d_2 = -1 \\ ([s,-,-,d_3],0,-1), \text{ for } a\ \varepsilon\ \{\$,\#\} \text{ and} \\ \qquad d_2 = 0 \\ s_r \text{ for } a = \$ \text{ and } d_2 = -1 \end{cases}$$

$$\text{d.}\quad \gamma([s,-,-,-,d_3],a) = \begin{cases} ([s,-,-,d_3],0,-1), \text{ for } a\ \varepsilon\ \Sigma \\ ([s,-,-,-],d_3), \text{ for } a = * \end{cases}$$

It is easily seen from the construction that A simulates C by carrying out the following procedure:

1. Drop a marker * to indicate the position of the reading head of C or the input tape. (This is done by a.)

2. Move left looking for the left boundary marker ¢ or markers of type stacked on ¢ to simulate the action of the first counter of C. (This is done by b.)

3. Move right to boundary to simulate the action of the second counter of C. (This is done by c.)

4. Move left to locate the * marker and repeat the process. (This is done by d.)

Conversely, suppose that an ∞-marker automaton A is given. We can construct a Turing machine T with a semi-infinite tape that simulates A as follows:

1. T divides its tape into chunks of identical length equal to the length of its input.

2. A stack of k markers on the jth square of the input tape of A can be simulated by T by writing a special symbol on the jth square of ith chunk for $1 < i < k$.

Since the quintuple description of T is very long, but relatively easy to construct, we will omit it here. ||

Definition 7. An ∞-marker automaton A is called linear bounded if the number of markers A is allowed to use is bounded by a linear function of the length of its input tape.

Theorem 6. Every linear-bounded automaton is equivalent to a linear-bounded marker automaton and vice versa.

Proof: Let $A = [S, \Sigma, \Gamma, \delta, s_o, F]$ be a given linear-bounded automaton such that Γ contains n elements. We assign an integer $1 \leq i \leq n$ to each element in Γ using a one-to-one function f. Then a marker-automaton B can simulate A by chopping or removing a stack of i markers to simulate the writing or erasing of a symbol $\alpha \in \Gamma$ by A, where α corresponds to the integer i. It is easily seen that, for a given input tape x, B needs to use at most $n\ell$ markers, when ℓ is the length of x, and hence is linear bounded. The construction of B is now given.

Let $B = [S', \Sigma, \gamma, s_o, s_a, s_r]$, where

$S' = S \cup \{s_a, s_r\} \cup \{[s,i,d] \mid s \in S \cup \bar{S}, 1 \leq i \leq n, d \in \{-1,0,1\}\}$

and $\bar{S} = \{\bar{s} \mid s \in S\}$.

Now γ is defined as follows:

$$\gamma(s,a) = \begin{cases} (t,\theta,d), \text{ if } (t,a,d) \in \delta(s,a) \text{ and } a \in \Sigma \\ ([t,f(p),d],0,0), \text{ if } (t,p,d) \in \delta(s,a) \text{ and } a \in \Sigma,\ p \in \Gamma \end{cases}$$

$$\gamma([s,i,d],a) = \begin{cases} ([s,i-1,d],*,0), \text{ if } i \geq 1 \text{ and } a \in \Sigma \cup \{*\} \\ (s,0,d), \text{ if } i=0 \text{ and } a \in \Sigma \cup \{*\} \end{cases}$$

$$\gamma(s,*) = ([\bar{s},1,0],-1,0)$$

$$\gamma([\bar{s},i,0]*) = ([\bar{s},i+1,0],-1,0)$$

$$\gamma([\bar{s},i,0],a) = ([\bar{s},i,0],0,0) \qquad a \neq *$$

$$\gamma([\bar{s},i,0],a) = ([t,f(p),d],0,0) \text{ if } i=h(q) \text{ and } \delta(s,q) = (t,p,d)$$

where $q,p \in \Gamma$

It is clear that B defined above is equivalent to A.

Since the converse of the theorem can be proved by revising the arugments, its proof is omitted here. ||

IV. Marker Automata and Some Classes of Languages

In this section, we present three results: 1. the family of equal matrix languages is acceptable by marker automata; 2. the family of derivation bounded languages is acceptable by marker automata; 3. the set of primes is acceptable by a deterministic 3-marker automaton.

Definition 8. A grammar $G = (V_N, V_T, S, P)$ is said to be an equal matrix grammar of dimension k (EMG k) if (1) V_T is the terminal vocabulary. V_N consists of the initial symbol S and the rest nonterminal symbols in the form of k-tuples $(A_1, A_2, \ldots, A_k)$ where if $(A_1, A_2, \ldots, A_k)$ and $(B_1, B_2, \ldots, B_k)$ are two k-tuples then $\{A_1, A_2, \ldots, A_k\} \cap \{B_1, B_2, \ldots, B_k\} = \emptyset$, (2) P consists of the following types of matrix rules: 1. A set of initial matrix rules of the form $[s \to \Lambda]$ or $[s \to f_1A_1f_2A_2 \ldots f_kA_k]$ where $f_1, f_2, \ldots, f_k$ (possibly empty) are elements of V_T^* where Λ denotes the empty word; 2. a set of nonterminal equal matrix rules of the form

$$\begin{bmatrix} A_1 \to f_1A_1 \\ A_2 \to f_2A_2 \\ \vdots \\ A_k \to f_kA_k \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} A_1 \to f_1B_1 \\ A_2 \to f_2B_2 \\ \vdots \\ A_k \to f_kB_k \end{bmatrix} \qquad (3)$$

where $f_1, f_2, \ldots, f_k \in V_T^*$ possibly empty but not all of $f_1, f_2, \ldots, f_k$ are empty; and 3. a set of terminal equal matrix rules of the form

$$\begin{bmatrix} A_1 \to f_1 \\ A_2 \to f_2 \\ \vdots \\ A_k \to f_k \end{bmatrix} \qquad (4)$$

where $f_1, f_2, \ldots, f_k \in V_T^*$ possibly empty but not all of $f_1, f_2, \ldots, f_k$ are empty.

To apply an equal matrix rule to a string x means to apply all the rules of the equal matrix rule which form it, to x in the given order (i.e., row 1 first, row 2 next, etc., and at last row k). If all the productions cannot be applied, then we say that the equal matrix rule is not applicable to x.

The following result is straingtforward and hence we will omit its proof here.

Lemma 5. Every EMGk G is equivalent to a normal from EMGk $G' = (V'_{N'}, V_T, S', P')$ such that 1. if $S' \to f_1A_1f_2A_2 \ldots f_kA_k \in P'$, then $f_i \in V_T \cup \{\Lambda\}$, for $1 \leq i \leq k$ and 2. if a matrix rule of the form in (3) or (4) of definition 8 is in P', then $f_i (1 \leq i \leq k)$ is in $V_T \cup \{\Lambda\}$ but not all f_i's are empty.

We are ready to prove that any equal matrix language is a marker language. Since the formal proof of this result involves a very lengthy, but intuitively simple construction of marker automaton, we will only describe how the constructed marker automaton behaves on an input tape. The reader should have no problem constructing such an automaton from the description.

Theorem 7. Every language generated by an equal matrix grammar of dimension k is acceptable by a (2k-1)-marker automaton.

Proof: Let L be generated by an equal matrix grammar, $G = (V_N, V_T, S, P^*)$ of dimension k. Without loss of generality, we may assume that G is in normal form as stated in lemma 5. We now describe the behavior of a (2k-1)-marker automaton A which accepts L.

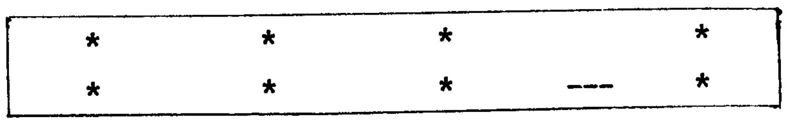
The state set of A consists of a number of "essential states" of the form $f_1A_1f_2A_2 \ldots f_kA_k(f_1\#_1f_2\#_2 \ldots f_k)$ if either $S \to f_1A_1 \ldots f_kA_k$, or

$$\begin{bmatrix} A_1 \rightarrow f_1A_1 \\ \vdots \\ A_k \rightarrow f_nA_n \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} B_1 \rightarrow f_1A_1 \\ \vdots \\ B_k \rightarrow f_kA_n \end{bmatrix} \quad \text{or}$$

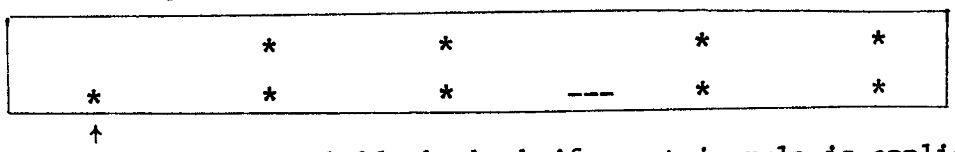
$$\begin{bmatrix} A_1 \rightarrow f_1 \\ \vdots \\ A_n \rightarrow f_n \end{bmatrix}$$

is a production in P*. These essential states are used to keep track of the derivations of G. A also has a number of "auxiliary states" to move markers on input tapes. The operation of A is now described in the following:

1. A partitions the input tapes nondeterministically into k blocks with k-1 pairs of markers. Each pair is stacked up on one square, as demonstrated by the following diagram. The bottom marker of each pair is used as place holder and hence will stay at that square during the entire process.



2. A places one marker on the left boundary end marker to the immediate left of the leftmost input symbol.

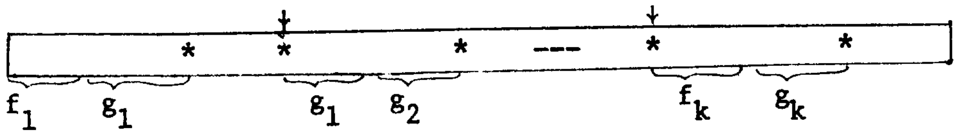


3. A uses one marker in each block check if a matrix rule is applied properly.

a) If $[S \rightarrow f_1A_1f_2A_2 \ldots f_kA_k]$ is an initial matrix rule, then one marker from the i-th block check if f is in the block for all $1 \leq i \leq k$. Since G is in normal form, this can be accomplished by moving the top marker of each stack one square to right and seeing if the symbol on the square where the marker moves is indeed f_i for $f_i \neq \Lambda$.

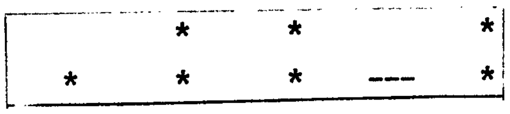
b) If $\begin{bmatrix} A_1 \rightarrow g_1B_1 \\ A_2 \rightarrow g_2B_2 \\ \cdots\cdot \\ A_k \rightarrow g_kB_k \end{bmatrix}$

is a matrix rule in P*, then A uses one marker from i-th block to check if g_i is the next terminal string in the block, for all $1 \leq i \leq k$. This is illustrated in the following diagram. Note that markers with one arrow pointing to them are place holders. A does not move these markers during the entire computation.



If in (a) or (b) any of the terminal string f_i is not found in the i-th block, then the entire matrix rule does not apply at that particular step. A matrix rule applies at a certain step only if all the f_i, $1 \leq i \leq k$, are found in all the k blocks and k markers (the left marker from each block) are moved to the end of the f_i, $1 \leq i \leq k$, in each block. If a matrix rule does not apply in a certain step, then the input is rejected in this round of attempt. (This does not mean the input is not acceptable by the marker automaton since the marker automaton since the marker automaton is nondeterministic.)

4. If all the k blocks are checked and the input is an element of L, then the final configuration of the markers on the tape should look like the following:



It is clear from the previous description of A that A accepts X if and only if $X \in L$. ||

Definition 9. A phrase-structure grammar $G = (V_N, V_T, S, P)$ is said to be in restricted normal form if the productions containing terminals in P are of the form: $u \to v$, where $u \in V_N$, $v \in V_T \cup V_TV_N \cup V_NV_T$.

Definition 10. A grammar G is said to be k-bounded if for each derivation $w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n$ in G, there are at most k elements of V_N occurring in each w_i $(1 \leq i \leq n)$. G is called derivation bounded if it is k-bounded for some k.

The following result is a simple consequence of the previous two definitions, and hence its proof will be omitted here.

Lemma 6. If G is a k-bounded grammar $(k \geq 1)$, then there is a k-bounded grammar G' in restricted normal form equivalent to G.

We are now ready to show that languages generated by derivation bounded grammars form a subclass of the family of marker languages. Again, only an informal sketch of the proof will be given for the following result for the same reason as in Theorem 7.

Theorem 8. Let L be a language generated by a k-bounded grammar. Then L can be accepted by a 2k-marker automaton.

Proof: By lemma 6, we may assume that the k-bounded grammar $G = (V_N, V_T, S, P)$ is in restricted normal form. The 2k-marker automaton A has a number of "essential states"

$$\{\alpha \mid \alpha \in \bigcup_{i=1}^{2k} \{ V_N \cup \{ * \}\}^i \text{, and at most k *'s and k elements of } V_N \text{ can occur in } \alpha\}$$

to keep track of the derivations of G, and a number of "auxiliary states" to move markers on input tape. The operation of the automaton is described in the following.

1. A begins in State S. If $S \to B_1B_2\ldots B_n$ is a production in P, then A goes into state $B_1B_2\ldots B_n$ while (nondeterministically) using (n-1) pairs of markers to partition its input tape into n blocks.

2. If A is in state α and B_i is the i-th element of V_N from left in α and if

a) $B_i \rightarrow C_1C_2...C_j$ is a production rule in P and β is the result of substituting $C_1...C_j$ for B_i in α, then A goes into state β while placing (j-1) pairs of markers between the (2i-1)th and 2ith markers originally on the tape. (This can be done since G is k-bounded.)

b) $B_i \longrightarrow aB_j$ is a rule in P, and either

i. B_i is not immediately preceded by a*, and β is the result of substituting $*B_j$ for B_i in α, or

ii. B_i is immediately preceded by the * symbol, and β is the result of substituting B_j for B_i in α,

then A goes into state β while moving the (2i-1)th marker one square to the right to check off an "a". It rejects the tape if "a" is not found. (Note that the last step for checking off a block successfully is when the (2i-1)th and 2ith markers are placed adjacently such that 2ith marker is on a square with input symbol a).

3. If $B_i \rightarrow B_ja$ is a production rule in P, then A goes through similar operations as described in 2(b).

4. If A is in state α such that there is a substring $A_iA_{i+1}...A_{i+p}$ in α and $A_i...A_{i+p} \rightarrow B_1B_2...B_j$ is a production rule in G, then A goes into state β, where β is the result of substituting $B_1...B_j$ for $A_i...A_{i+p}$ in α, and removes p pairs of markers after the (2i-1)th marker on the tape and fill in with (j-1) pairs of markers between the (2i-1)th marker and the 2ith marker (nondeterministically).

5. Finally, A accepts the input whenever it is in a state belonging to $\{*\}^i$, $1 \leq i \leq k$.

It should be clear from the above description that A accepts an input X if and only if $X \in L(G)$. ||

It is easily seen that the language $L_1 = \{a^n * a^{kn} \mid k, n \geq 1\}$ is a 2-marker acceptable language. The language $L_2 = \{a^n \mid n = p(i+1),\ i,p \geq 1\}$ is 3-marker acceptable by using one marker to act as * in L_1. Following this kind of reasoning, we see that one can use three markers to check if the length of a string of a's is a multiple of any number. This gives rise to the following result.

Theorem 9. The set of primes (in unary representation) is acceptable by a deterministic, halting 3-marker automaton.

The previous theorem, along with two interesting open problems, give motivation to study the 3-marker automaton in detail. It was conjectured by Harmanis and Shank [3] that linear bounded automaton is the weakest automaton which can recognize the set of primes. Since a 3-marker automaton recognizing the set of primes is bounded in time by a linear function of the length of its tape, it seems desirable to study the relationship between marker automata and classes of Turing machine defined by complexity measures. In particular, since we have shown by theorem 6 that Linear bounded automata is equivalent to linear bounded infinite marker automata, it would be desirable to obtain the precise bound of 3-marker automaton. Furthermore, we see that 3-marker automaton is much simpler than a linear bounded automaton in general.

Although we are not able to prove that the set of primes cannot be accepted by any 2-marker automaton, it is easy to see that the automaton will have to check whether the input number is divisible by any number other than 1 and itself. Intuitively, this task cannot be done by any 2-marker automaton. Based on these intuitive reasons, we conjecture that 3-marker automaton is the weakest automaton which can recognize the set of primes. Another good reason for studying 3-marker automaton is based on a result of Hartmanis [4] who showed that the open problem of equivalence of deterministic and nondeterministic linear bounded automata would be solved if languages accepted by a 3-head nondeterministic automaton can be accepted by a k-head deterministic automaton.

V. Concluding Remarks

In the previous sections, we have discussed some general properties of marker automata as well as their relationships to other types of automata and languages. Closure properties and decision problems were not discussed because they have been investigated elsewhere.

A most interesting result in our investigation is the hierarchy theorem of deterministic and deterministic, halting marker languages. Since it is easily shown that even 2-marker automata can recognize context-sensitive languages, this theorem gives a new classification of languages. Similar result has been obtained by Hartmanis [4] in dealing with multi-head automata. It remains an open problem as whether this result holds in the case of non-deterministic marker languages. It was shown by Hartmanis [4] and Savitch [10] that the relationship between deterministic and non-deterministic marker automata is directly related to the open problem as to whether deterministic and non-deterministic linear-bounded automata are equivalent. Indeed, the results of Hartmanis and Savitch, and Hartmanis and Shank [3] give motivation to study the characterization of 3-marker automata as mentioned at the end of the previous section.

VI. Bibliography

1. Blum, M., and C. Hewitt. "Automata on a 2-dimensional tape," _IEEE Conference Record of 1967 Eighth Annual Symposium on Switching and Automata Theory_, 155-160.

2. Hartmanis, J. and H. Shank. "Two memory bounds for the recognition of primes by automata," _Mathematical Systems Theory_, Vol. 3 (1969), 125-129.

3. ______________________. "On the recognition of primes by automata," _Journal of the ACM_, Vol. 15 (1968), 382-389.

4. Hartmanis, J. "On Non-Determinacy in Simple Computing Devices," _AcTa Information_ (1972).

5. Hsia, P. and R.T. Yeh. "Marker Automata," _Proc. International Symposium on Theory of Automata and Programming Languages_ (1972).

6. Kreider, D. L., and R. W. Ritchie. "A basis theorem for a class of two-way automata," _Zeitshr, Fur Math. Logik und Grundlagen der Math_, Bd. 12 (1966), 243-255.

7. ______________________. "A universal two-way automaton," _Archiv Fur Mathematishe Logik und Grundlagenforschung_, (1966), 49-58.

8. Ritchie, R. W., and F. N. Springsteel. "Language Recognition by Marking Automata," _Information and Control_ 20 (1972), 313-330.

9. Rosenberg, A. L. _Nonwriting Extensions of Finite Automata_, Doctoral Thesis (1965), Harvard University, Cambridge, Mass.

10. Savitch, W. J. "Non-deterministic Finite Automata Revisited," _Proc. Sixth Hawaii International Conference on System Sciences_ (1973), 249-251.

11. Siromoney, R. "On equal matrix languages," _Informatica and Control_, Vol. 14 (1969), 135-151.

12. Springsteel, F. N. _Context-Free Languages and Marking Automata_, Doctoral Thesis (1967), University of Washington, Seattle, Washington.