Theoretical Computer Science 2 (1976) 271–294.
© North-Holland Publishing Company

FINITE AUTOMATA WITH MULTIPLICATION*

Oscar H. IBARRA, Sartaj K. SAHNI and Chul E. KIM[1]

Department of Computer Science, University of Minnesota, Minneapolis, MN55455, U.S.A.

Communicated by M. Harrison
Received 22 January 1975
Revised 3 July 1975

Abstract. A finite automaton with multiplication (FAM) is a finite automaton with a register which is capable of holding any positive rational number. The register can be multiplied by any of a fixed number of rationals and can be tested for value 1. Closure properties and decision problems for various types of FAM's (e.g. two-way, one-way, nondeterministic, deterministic) are investigated. In particular, it is shown that the languages recognized by two-way deterministic FAM's are of tape complexity log n and time complexity n^3. Some decision problems related to vector addition systems are also studied.

0. Introduction

It is well known that a one-way finite automaton with two counters can recognize any recursively enumerable set [12]. Clearly, the two counters can be simulated by a single register that is capable of storing any positive rational number whose value can be modified and tested as follows: The register can be multiplied by any of a fixed number of rational numbers and, in addition, the machine can check if any of these rationals is a factor of the register value. If p/q and r/s are two rationals in lowest terms, then p/q is a factor of r/s iff p divides r and q divides s. For example, if the register value is 6/35, then 2/5 is a factor, but 3/25 is not. Such a machine, similar to the program-machine of [12], is then equivalent to a Turing machine. Thus, we see that multiplication and "factor test" are sufficient operations for single-register machines to recognize all recursively enumerable sets. As one might expect, a weaker set of operations reduces the computing power of these machines. One such device is studied in this paper: the program-machine as described above except that the "factor test" is replaced by an "equality to 1" test (i.e., the register can only be tested for being equal to 1). We call such a machine a finite automaton with multiplication (FAM). A FAM is considerably weaker than a Turing machine. In fact, the languages recognized by two-way deterministic FAM's belong to the

* The work of the first author was supported in part by National Science Foundation Grant DCR 72-03728-A01, and the work of the second author was supported in part by the University of Minnesota Graduate School Research Grant 481-0906-4125-02 and by NSF Grant DCR 74-10081.
[1] Present address: University of Maryland, College Park, MD 20742, U.S.A.

tape-complexity class log n and time-complexity class n^3. One may readily verify that a FAM is equivalent to a multicounter automaton which can only detect when its counters are simultaneously zero. We shall also see that the notion of FAM's is useful in studying certain decision problems concerning vector addition systems and integer programming.

The paper is divided into 5 sections. In Section 1, we give definitions and notations we use throughout the paper. The two-way nondeterministic FAM is defined formally as a language recognizer where the mode of acceptance is by accepting state with the register value equal to 1. Special classes of FAM's are then introduced and some examples are given to illustrate the workings of these automata.

In Section 2 we show that the languages recognized by two-way deterministic FAM's are recognizable by deterministic Turing machines of tape-complexity log n and time-complexity n^3. Closure properties of one-way nondeterministic and deterministic FAM's are then investigated. In particular, it is shown that the languages recognized by one-way deterministic FAM's are incomparable with the context-free languages nor do they form an abstract family of languages (AFL) [3]. The one-way nondeterministic languages, however, form an AFL.

In Section 3 we consider FAM's in which the "equality to 1" is not allowed as a primitive operation. We call those machines FAM's without equality. Contrasting closure properties are obtained (e.g., the languages recognized by the one-way nondeterministic such machines do not form an AFL). We also show that the class of bounded languages recognized by one-way nondeterministic FAM's without equality is precisely the class of bounded semilinear languages as defined in [6]. A class of machines similar to FAM's without equality called multiplicative analog memory automata (MAMA) was recently introduced in [17]. The only difference is that in a MAMA, the register (called analog in [17]) has a cut-off value of 1 (i.e., the device halts and rejects the input whenever the register value exceeds 1). The computing capability of the MAMA, however, was not investigated in [17].

Section 4 gives inclusion relations among the different classes of FAM's and discusses some decidability questions. It is shown that the emptiness, containment and equivalence problems for one-way deterministic FAM's without equality are solvable while the universe problem for the nondeterministic case is undecidable.

We conclude the paper in Section 5 where we summarize our results and state a few open problems.

1. Preliminaries

We begin by defining the general model of a FAM.

Definition. *A two-way nondeterministic finite automaton with multiplication* (2NFAM) is a device $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F\rangle$, where $K, \Sigma, F \subseteq K, \Gamma$ are finite sets of states, inputs, accepting states, and positive rational numbers (called *multipliers*), q_0 in K is the initial state, ¢ and \$ not in Σ are left and right endmarkers for the inputs, and δ is a mapping from $K \times (\Sigma \cup \{¢, \$\}) \times \{=, \neq\}$ into the set of all subsets of $K \times \{-1, 0, +1\} \times \Gamma$.

The 2NFAM M has an input tape of the form $¢ a_2 \cdots a_{n-1} \$$ and a register which can store any positive rational number, its initial value being equal to 1. (p, d, m) in $\delta(q, a, \alpha)$ (where q, p are in K, a in $\Sigma \cup \{¢, \$\}$, α in $\{=, \neq\}$, and d in $\{-1, 0, +1\}$) means that if M is in state q, reading the symbol "a" on the input tape and the register has some value y, where $y = 1$ or $y \neq 1$, then M may change its state to p, move its input head as specified by d (i.e., move one square left, no move, move 1 square right if $d = -1, 0$ or $+1$, respectively) and multiply its register by m.

Let x be a string in Σ^*.[2] An instantaneous description of M operating on the input tape $¢x\$$ can be described by a 4-tuple $(q, ¢x\$, i, y)$, where q is a state of M, $¢x\$$ is the input tape, i is the position of the input head within $¢x\$$ $(1 \leq i \leq |¢x\$|)$ and y is the value of the register.[3] A move of M can be indicated by the relation $\vdash$ between instantaneous descriptions. Thus $(q, ¢x\$, i, y) \vdash (p, ¢x\$, j, w)$ if M can attain instantaneous description $(p, ¢x\$, j, w)$ from $(q, ¢x\$, i, y)$ in one move. The reflexive-transitive closure of $\vdash$ is denoted by $\vdash^*$.

A string x in Σ^* is accepted by M if $(q_0, ¢x\$, 1, 1) \vdash^* (q, ¢x\$, |¢x\$|, 1)$ for some q in F (i.e., M started in its initial state q_0 with the input head on ¢ and register value equal to 1 eventually enters an accepting state with the input head on the right endmarker and register value equal to 1). We assume that $\delta(q, \$, x) = \emptyset$ for all q in F, i.e., in an accepting state M always halts on \$. We shall also assume that M is prevented from falling off either end of the input tape. The set of strings (i.e., the language) *accepted* (or *recognized*) by M is the set

$$T(M) = \{x \mid (q_0, ¢x\$, 1, 1) \vdash^* (q, ¢x\$, |¢x\$|, 1) \text{ for some } q \text{ in } F\}.$$

A 2NFAM $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F \rangle$ is *deterministic* (2DFAM) if $|\delta(q, a, \alpha)| \leqq 1$ for each q in K, a in $\Sigma \cup \{¢, \$\}$, and α in $\{=, \neq\}$.[4] M is one-way (denoted by 1NFAM) if (p, d, m) in $\delta(q, a, \alpha)$ implies $d \geqq 0$.

A *2NFAM without equality* (denoted by 2NFAMW) is a 2NFAM $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F \rangle$ with the property that $\delta(q, a, \neq) = \delta(q, a, =)$ for all q in K and a in $\Sigma \cup \{c, \$\}$ (i.e., M cannot detect whether or not the register has value 1). In this case, we may omit writing the third component of the domain in describing the mapping δ.

We shall use the abbreviations 2NFAM, 2DFAM, 1NFAM, 1DFAM,

[2] Σ^* is the set of all finite strings of symbols in Σ, including the null string denoted by ε. If x and y are strings in Σ^*, xy is the string x followed by the string y; $x^0 = \varepsilon$, and $x^{k+1} = x^k x$ for all $k \geqslant 0$.

[3] $|z|$ denotes the length of z (i.e., the number of symbols in z).

[4] If S is a finite set, then $|S|$ denotes the cardinality of S.

2NFAMW, 2DFAMW, 1NFAMW, 1DFAMW to denote the different types of FAM's described above. For example, 1DFAMW refers to a one-way deterministic FAM without equality.

By $\mathscr{L}$(1NFAM), $\mathscr{L}$(1DFAM), etc., we shall mean the set of all languages recognized by the corresponding FAM.

We say that a FAM M (of any type) operates in time $t(n)$ if for any input x, $|¢x\$| = n$, that is accepted, M has a sequence of at most $t(n)$ moves leading to the acceptance of x.[5]

To illustrate the working of these automata we describe below in Examples 1.1 and 1.2, FAM's , M_1 and M_2, such that $T(M_1) = \{a^n b^n c^n \mid n \geq 0\}$ and $T(M_2) = \{a^n b^m \mid m \geq n \geq 0\}$.

Example 1.1. $M_1 = \langle K_1, \Sigma_1, \Gamma_1, \delta_1, q_{01}, ¢, \$, F_1\rangle$, where $K_1 = \{q_1, q_2, q_3, q_4\}$, $\Sigma_1 = \{a, b, c\}$, $\Gamma_1 = \{\frac{1}{6}, 2, 3\}$, $q_{01} = q_1$, $F_1 = \{q_4\}$, and δ_1 is defined as follows: $\delta_1(q_1, ¢) = (q_1, +1, 1)$, $\delta_1(q_1, a) = (q_1, +1, \frac{1}{6})$, $\delta_1(q_1, \$) = (q_4, 0, 1)$, $\delta_1(q_1, b) = (q_2, +1, 2)$, $\delta_1(q_2, b) = (q_2, +1, 2)$, $\delta_1(q_2, c) = (q_3, +1, 3)$, $\delta_1(q_3, c) = (q_3, +1, 3)$, $\delta_1(q_3, \$) = (q_4, 0, 1)$.

M_1 is a 1DFAMW so that (by convention) "$=$" and "$\neq$" have been omitted from the move function δ_1. M_1 goes through the input and multiplies the register by $\frac{1}{6}$, 2, or 3 depending on whether it sees an a, b, or c. Thus, if the input is of the form $a^i b^j c^k$, the register value at the end of the process is $v = (\frac{1}{6})^i \cdot 2^j \cdot 3^k$. Clearly, $v = 1$ if and only if $i = j = k$. Also, the finite state control rejects inputs not of the form $a^i b^j c^k$. It follows that $T(M_1) = \{a^n b^n c^n \mid n \geq 0\}$.

Example 1.2. $M_2 = \langle K_2, \Sigma_2, \Gamma_2, \delta_2, q_{02}, ¢, \$, F_2\rangle$, where $K_2 = \{q_1, q_2, q_3\}$, $\Sigma_2 = \{a, b\}$, $\Gamma_2 = \{1, \frac{1}{2}, 2\}$, $q_{02} = q_1$, $F = \{q_3\}$, and δ_2 is defined as follows: $\delta_2(q_1, ¢, =) = (q_1, +1, 1)$, $\delta_2(q_1, \$, =) = (q_3, 0, 1)$, $\delta_2(q_1, a, =) = (q_1, +1, \frac{1}{2})$, $\delta_2(q_1, a, \neq) = (q_1, +1, \frac{1}{2})$, $\delta_2(q_1, b, =) = (q_3, +1, 1)$, $\delta_2(q_1, b, \neq) = (q_2, +1, 2)$, $\delta_2(q_2, b, \neq) = (q_2, +1, 2)$, $\delta_2(q_2, b, =) = (q_3, +1, 1)$, $\delta_2(q_2, \$, =) = (q_3, 0, 1)$, $\delta_2(q_3, b, =) = (q_3, +1, 1)$.

M_2 with input of the form $a^n b^m$ goes through the "a" segment and multiplies the register by $\frac{1}{2}$ for each a it sees. Thus, at the end of the "a" segment, the register value would be $(\frac{1}{2})^n$. Then M_2 multiplies the register by 2 for each b in the first n b's it sees, thereafter multiplying the register by 1. It should be clear that $T(M_2) = \{a^n b^m \mid m \geq n \geq 0\}$.

Remarks. (a) A 1DFAM with $\Gamma = \{\frac{1}{2}, 2\}$ can simulate any one counter machine (see [8] for a definition of counter machines). It follows from the result of Minsky [12] that a 1DFAM with two registers is equivalent to a Turing machine.

(b) $\{a^n b^n c^n \mid n \geq 0\}$ is clearly recognizable by a 1DFAMW but is not a context-

[5] We assume that $t(n)$ is a computable function from positive integers into positive integers.

free language [8] and so not recognizable by a one-way one-counter machine. $\{a^n b^n \mid n \geq 0\}^*$ is a deterministic context-free language [2] recognizable by a one-way one-counter machine but is not 1NFAMW recognizable. (The proof is given in Lemma 3.1.) It follows that $\mathscr{L}$(1DFAMW) and $\mathscr{L}$(1NFAMW) are incomparable with context-free languages, deterministic context-free languages, and languages recognizable by one-way one-counter machines.

(c) Any type of FAMW with several registers can easily be converted to an equivalent FAMW of the same type with only one register (by choosing distinct primes for the multipliers). Thus, there is no complexity hierarchy based on the number of registers.

2. Finite automata with multiplication

In this section, we investigate the properties of FAM's. We begin by showing in Section 2.1 that languages in $\mathscr{L}$(2DFAM) belong to the tape complexity class $\log(n)$ and time complexity class n^3. Then, in Section 2.2 we exhibit languages that are not in $\mathscr{L}$(1FAM). These results are then used in Section 2.3 to obtain the closure properties of 1FAM's. Our proofs involve the notion of *cycle*. By a cycle of M we shall mean a sequence of moves starting and ending in the same state q_i. The *length* of a cycle is the number of input symbols over which M has moved since it was last in the same state of that cycle. The computation of M on reaching the endmarker \$ will consist of some number (≥ 0) of cycles followed by an incomplete cycle.

2.1. *Relationship to* $\log(n)$*-tape Turing machines*

We begin with the following lemma.

Lemma 2.1. *Let* $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F\rangle$ *be a 1DFAM. Then M operates in time* $t(n) \leq c \cdot s^3 \cdot n$ *(c is a constant depending only on* Γ *and s is the number of states of* M*) for all strings* $x \in T(M)$, $|¢x\$| = n$.

Proof. Let $P = \{p_1, \cdots, p_s\}$ be the primes appearing in the prime decomposition of the numerators and denominators of the multipliers, $\Gamma = \{m_1, m_2, \cdots, m_t\}$, of M. For $i = 1, \cdots, t$, let $m_i = \Pi_{j=1}^{s} p_j^{e_{ij}}$ where e_{ij}'s are integers. Let $e = \max_{i,j} |e_{ij}|$.

If a string x is accepted, the read head eventually moves right from every symbol, except the \$, of the input tape $¢x\$ = a_1 a_2 \cdots a_n$. If the register does not become 1 on a_i, M moves its head to a_{i+1} after at most $|s|$ stationary moves. If the register does become 1, then it can become 1 at most $|s|$ times, as otherwise M enters an infinite cycle on a_i.

Suppose that the register value becomes 1 on both a_i and a_j, $j < i$, and not on a_k for any $j < k < i$.

Examine the computation of M between times when the register value becomes 1 for the last time on a_j and for the first time on a_i. The value of the register, α, when the read head moves into a_i depends only on computations carried out on $(i-j)$ symbols $a_j\, a_{j+1} \cdots a_{i-1}$ (on a_j, the computation after the register became 1 for the last time). Since at most $|s|$ moves are possible on each of these, the largest possible absolute value for the exponent of the primes in the prime decomposition of the numerator and denominator of α is $e*(i-j)*|s|$. On a_i the computation of M until the register becomes 1 is as follows: M enters a_i in state q with register value α. Then, it is in state q' with register value $\alpha\beta$ after less than $|s|$ moves, when it enters a cycle. For each of these cycles M multiplies the register by γ. Finally, after an incomplete cycle with a multiplication of δ the register becomes 1, $\alpha\beta\gamma^r\delta = 1$.

$$q \overset{\beta}{\cdots} \underline{q' \overset{\gamma}{\cdots} q'} \underline{\overset{\gamma}{\cdots} q'} \underline{\cdots q'} \underline{\overset{\gamma}{\cdots} q'} \overset{\delta}{\cdots} q''$$

β and δ involve at most $|s|$ multipliers each and so the largest absolute value of exponents of prime factors of numerator and denominator of $\alpha\beta\delta$ is at most $e*(i-j+2)*|s|$. For $\alpha\beta\gamma^r\delta$ to equal 1 for some integer r, every prime factor in $\alpha\beta\delta$ should be cancelled by the corresponding prime factors in γ^r and every prime factor in γ changes the exponent of the corresponding prime in $\alpha\beta\delta$ by at least 1 for each cycle, which is of length at most $|s|$. Thus $r \leq e*(i-j+2)*|s|^2$. If the register becomes 1 on a_i more than once, the computation of M between consecutive 1's takes less than $2*e*|s|^2$, since it is identical to the one above except that the register value is 1 instead of α.

Hence, on a_i on which the register value becomes 1, the computing time is bounded by

$$e*(i-j+2)*|s|^3 + 2*e*|s|^2*|s| = e*|s|^3*((i-j)+4).$$

The total computing time for $¢x\$$ is then bounded by

$$\sum_{i\notin I}^{n} |s| + \sum_{i\in I} e\,|s|^3((i-j)+4) \leq 6e\,|s|^3 n,$$

where I is the set of all integers i such that on a_i the register value becomes 1. $\square$

Corollary 2.1. *Let* $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F\rangle$ *be a 2DFAM. Then M operates in time* $O(n^3)$.

Proof. Given M and an input x with $|¢x\$| = n$, we construct a 1DFAM M_x with the property that M_x operates on input $¢\$$ in t_x steps if and only if M operates on input $¢x\$$ in t_x steps.

Moreover, M_x will have $n \cdot |s|$ states. It would then follow from Lemma 2.1 that M_x on input $¢\$$ operates in $\leq c'(n\cdot|s|)^3$ steps for some constant c'. M_x is constructed as follows: $M_x = \langle K_x, \Sigma, \Gamma, \delta_x, q_{0x}, ¢, \$, F_x \rangle$, where $K_x = \{(q, ¢x\$, i) \mid q$ in $K, 1 \leq i \leq |¢x\$|\}$, $q_{0x} = (q_0, ¢x\$, 1)$, $F_x = \{(q, ¢x\$, |¢x\$|) \mid q$ in $F\}$, and δ_x is defined by:

(1) $\delta_x((q_0, ¢x\$, i), ¢, =) = ((q_0, ¢x\$, 1), 1, 1)$.

(2) For each q in K, $1 \leq i \leq |¢x\$|$, α in $\{=, \neq\}$, $\delta_x((q, ¢x\$, i), \$, \alpha) = ((p, ¢x\$, i+d), 0, m)$ if $\delta(q, a_i, \alpha) = (p, d, m)$, where a_i is the ith symbol of $¢x\$$.

M_x encodes the input $¢x\$$ in its finite control and simulates M in the finite control. It is obvious that M_x has $n \cdot |s|$ states and that M_x accepts $¢x\$$ in t_x steps. □

Theorem 2.1. *If M is a 2NFAM (2DFAM) operating in polynomial time, then $T(M)$ is accepted by a* log n *tape-bounded nondeterministic (deterministic) Turing machine.*

Proof.[6] Let $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F \rangle$. Let $p_1, \cdots, p_k$ be the prime numbers appearing in the prime decompositions of the numerators and denominators of multipliers in Γ. Then during the computation of M on $¢x\$$, the value of its register takes the form $p_1^{i_1} \cdots p_k^{i_k}$, where $i_1, \cdots, i_k$ are integers (positive, negative, or zero). At the start $i_1 = i_2 = \cdots = i_k = 0$. Since M operates in polynomial time, the absolute value of i_l $(1 \leq l \leq k)$ is at most cn^r for some integers c and r, where $n = |¢x\$|$. It is obvious that a log n tape-bounded nondeterministic (deterministic) Turing machine can keep track of the values of $i_1, \cdots, i_k$, and hence the simulation of M is possible. □

Corollary 2.1 taken together with the above theorem gives:

Corollary 2.2. *If M is a 2DFAM, then $T(M)$ is of tape-complexity* log n.

Corollary 2.3. *If M is a 2DFAM, then $T(M)$ can be accepted by a deterministic Turing machine of time-complexity n^3.*

Proof. This follows from Corollary 2.1 and the observation that the Turing machine constructed in the proof of Theorem 2.1 can be made to operate in time n^3 if it is given n^3-space. □

2.2. Languages not in $\mathscr{L}$(1DFAM)

Here, we show that certain languages are not 1DFAM recognizable. These results will be used in Section 2.3 to obtain the closure properties of 1FAM's. Since we are

[6] See [8] for a discussion of tape and time bounded Turing machines.

unable to obtain any general pumping lemma for FAM's, it will be necessary to give individual proofs for each of the languages we consider.

Lemma 2.2. $L_1 = \{a^nb^n \mid n \geq 0\} \cup \{a^nb^{2n} \mid n \geq 0\} \notin \mathscr{L}(1DFAM)$.

Proof. The proof is carried out in three steps. First, in step 1, we show that for any 1DFAM, M, recognizing L_1, there exists a constant, c, such that if the value of the register is equal to 1 at any time during M's computation on the segment of b's of $x \in L_1$ then $d(n) \leq c$ ($d(n)$ is the distance, i.e., the number of input symbols, of M's read head from the right endmarker, \$, when the register first has the value 1 on the b's). Then, we show that this distance, $d(n)$, must in fact be zero except for a finite number of values of n for which $d(n)$ may be > 0. Finally, in step 3, we show that L_1 is not recognizable by any M for which the register never has the value 1 on the b's (except on a finite number of n's).

Step 1. Assume that M recognizes L_1 and $d(n)$ is not bounded by any constant. Then, there are at least $s+1$ values of n, $(n_1, n_2, \cdots, n_{s+1})$ such that $d(n_i) \neq d(n_j)$, $i \neq j$ (s is the number of states of M). Consider the state M is in when the register first becomes 1 on the segment of b's for each of these n's. Since M has only s distinct states, it must be the case that for two n's, n_i and n_j $(i \neq j)$, M is in the same state and reading the same symbol "b". But $d(n_i) \neq d(n_j)$. We may assume without loss of generality that $d(n_j) > d(n_i)$. Since M is deterministic, M must accept $a^{n_i}b^{2n_i + d(n_j) - d(n_i)}$ which is not a member of L_1. Consequently, $d(n) \leq c$ for some constant c.

Step 2. $d(n) = 0$ except for a finite number of n's. Suppose that $d(n) \neq 0$ for infinitely many n's. Denote by A_1 and A_2 the languages $\{a^nb^n \mid n \geq c\}$ and $\{a^nb^{2n} \mid n \geq 0\}$, respectively. Since $A_2 \subsetneqq L_1$, $d(n) \leq c$ and M is a 1DFAM, it follows that for $x \in \{a^nb^n \mid n > c\}$, $d(n) = 0$ as otherwise there is an m such that $x = a^mb^{2m}$ amd $d(m) > m > c$. Hence, there must be infinitely many n's for which $x = a^nb^{2n}$ and $d(n) \neq 0$. We shall show that there is an infinite subset $\{r_1, r_2, \cdots\}$ of these n's for which

(a) if $x = a^{r_i}b^{2r_i}$ and $y = a^{j}b^{2r_j}$ then M is in the same state q_l at $d(r_i)$ and at $d(r_j)$,

(b) for all $x \in \{a^{r_i}b^{r_i}\} \cup \{a^{r_i}b^{2r_i}\}$, M enters the segment of b's following the a's in the same state,

(c) for all $x \in \{a^{r_i}b^{r_i}\}$, M enters the right endmarker, \$, in the same state q_i and exits the cycle on the \$ in the same q_j.

To see this, start with the infinite set of n's for which $d(n) \neq 0$ when $x = a^nb^{2n}$. Since the number of states in M is finite, an infinite subset of these n's must satisfy (a). An infinite subset of these n's must satisfy (b) as M is finite and deterministic. The same argument implies that there is an infinite subset $\{r_1, r_2, \cdots\}$ of these n's that satisfies (c) also.

Consider the behavior of M on input $x_i = a^{r_i}b^{r_i}$ for r_i, one of the r's described above. Since M must set its register to 1 in order to accept x_i, let us assume it multiplies the register by $k_0\, p^{j_i}\, k_1$, where p represents the multiplication within a cycle, on the \$, k_0 the multiplications made before M entered the cycle on the \$, and k_1 the multiplicand contributed by the incomplete cycle made before M left the cycle. Note that k_0, p and k_1 do not depend on r_i as we have fixed the state in which M can enter the \$ as well as the state in which M leaves the cycle. We also note that for two distinct r's r_i and r_l, the number of times the cycle on the \$ is made (i.e., j_i and j_l) is different (as otherwise M will also accept $a^{r_i}b^{r_i+r_j}$ and $a^{r_j}b^{r_i+r_j}$, since $j_i = j_l$ implies that register has the same value and state after $a^{r_i}b^{r_i}$ as after $a^{r_j}b^{r_j}$). Note that since $0 < d(r_i) \leqslant c$ and $r_i \geqslant \max\{c, s\}$, the moves of M form cycles on the b's. The length, $\lambda \leqslant s$, of each cycle is the same as M's register is $\neq 1$ until M's read head gets to $d(r_i)$. The amount by which the register is changed in each cycle is δ (i.e., the multiplicative change). $\delta \neq 1$ as otherwise deletion of $\lambda \leqslant s$ b's would result in an x that is also accepted by M. The value of the register at the end of the first r_i b's is $(k_0 p^{j_i} k_1)^{-1}$ for the input $x = a^{r_i}b^{2r_i}$. But at $d(r_i)$ the register becomes 1. Therefore $\delta^{f(r_i)}k_2 = k_0 p^{j_i} k_1$, where $f(r_i) = \lfloor (r_i - d(r_i))/\lambda \rfloor$ and k_2 is the contribution of the unfinished cycle segment on the last r_i b's. Again, note that k_2 does not depend on r_i as for all r, M enters the last r b's in the same state and also reaches $d(r)$ in the same state.

We now obtain the following:

$$\delta^{f(r_i)}k_2 = k_0 p^{j_i} k_1, \qquad \delta^{f(r_l)}k_2 = k_0 p^{j_l} k_1;$$

dividing out, we get

$$\delta^{m_1} = p^{m_2}, \quad \text{where} \quad m_1 = f(r_i) - f(r_l) \quad \text{and} \quad m_2 = j_i - j_l.$$

Further, since it must be that $j_i > j_l$ for some $r_i > r_l$, we assume $m_1, m_2 > 0$. However, this implies that M accepts all inputs of the form $a^n b^{n+\lambda m_1}$, $\lambda m_1 < n$, for n, one of the infinitely many r's.

This contradicts our assumption that $T(M) = L_1$. Hence $d(n) = 0$ except for infinitely many n.

Step 3. Since $d(n) = 0$ and M is deterministic, it follows that for infinitely many k, M enters the right endmarker, \$, in the same state on the inputs $x_1 = a^{kr}b^{kr}$ and $x_2 = a^{kr}b^{2kr}$ where $r = s!$ (this is so as the cycle length $\leqslant s$). If v_1 and v_2 are the values of the register on entering the \$, then $v_2 = v_1\delta^{kr/\lambda}$ (λ is the cycle length on the b's and δ the effective multiplication per cycle). Repeating the argument of step 2 this leads us to conclude that $\delta^{m_1} = p^{m_2}$ for some $m_1, m_2 > 0$ implying that $T(M) \neq L_1$.

Consequently, there is no M such that $T(M) = L_1$. □

Lemma 2.3. $L_2 = \{b^n(a^n b^n)^k \mid n \geqslant 1, k \geqslant 1\} \notin \mathscr{L}(\text{1DFAM})$.

Proof. Let M be a 1DFAM recognizing L_2. Then, M's register cannot be equal to 1 anywhere after the first b^n segment and before the right endmarker, \$, except for a finite number of n's. Suppose this is not the case. Then, for infinitely many n's, M's register becomes 1 either on an "a" or on a "b". Without loss of generality, let us assume this happens on a "b" for infinitely many n's. Since M has only s distinct states, M is in the same state and reading the same symbol "b" for infinitely many n's when its register becomes 1. Interchanging the tails of these inputs (if the tail does not contain at least one complete $a^n b^n$ segment, just consider a bigger k value for the same n) results in the acceptance of strings not in L_2.

Define an entering state to be a state in which M moves to a new symbol. For n large enough, M must repeat some entering state on the segment of a's. By the preceding argument M's register is not equal to 1 on the a's. Hence, there are at most s distinct cycles that M can go through on the a's. Consequently, for $k > s$, M must repeat the same cycle on the a's on two different $a^n b^n$ segments. Let λ be the length of this cycle and let δ_λ be the multiplicative change of the register. δ_λ is a constant depending only on the specifications of M. We claim that for k sufficiently large there are two such $a^n b^n$ segments towards the right end of the input (we choose the closest such segments) such that moving a's from one segment to the other allows M to perform the same computation. Such a change does not alter the value of the register nor M's state when M reaches the \$. Hence, the string so obtained would be accepted by M. But, this string is not a member of L_2. Consequently $T(M_2) \neq L_2$. It follows that $L_2 \notin \mathcal{L}(\text{1DFAM})$.

Note that our claim will be true if the absolute value of the exponent of at least one of the primes in the decomposition $(e_1, e_2, \cdots, e_r)$ of the register value (register value $= p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$) becomes arbitrarily large. Then, for at least one e_i, $|e_i|$ is greater than the exponents contributed by M between the two segments (for large enough k). As a result, moving λ a's cannot result in the register becoming 1 on the new input segment. Let us now prove that $|e_i|$ can be made arbitrarily large for some e_i.

Look at the states, $(q'_1, q'_2, \cdots, q'_k)$, M is in at the end of each $a^n b^n$ segment. For $k \gg s$, there must be some repetitive cycle in $(q'_1, q'_2, \cdots, q'_k)$, i.e., it is of the form $(q'_1, q'_2, \cdots, q'_j, Q^i, q'_l, q'_{l+1}, \cdots, q'_k)$ where j, $(k - l + 1) < s$ and each Q represents a block of states $q_{m_1} q_{m_2} \cdots q_{m_r}$, $r < s$, $q_{m_i} \neq q'_p$, $1 \leq p \leq j$, $l \leq p \leq k$. Note that Q is fixed since M's register is never 1 after the first b^n segments and M is deterministic. Let δ_n be the multiplicative factor by which M's register changes on each Q segment. Except for a finite number of n's, $\delta_n \neq 1$. This follows from the observation that if $\delta_n = 1$ for infinitely many n's, then for two such n's $n_1 \neq n_2$, the Q's are formed by the same state cycles. Since the input x read during the Q segment of n_1 contains at least one occurrence of $a^{n_1} b^{n_1}$, substituting this input for the input y read during the Q segment of n_2 results in a string which is not in L_2 but is accepted by M.

Hence, for any constant, c, the absolute value of the exponents of at least one of the primes in the decomposition of the register value becomes $\geq c$ when M's read head is a distance d_c from the left endmarker ¢. Note that d_c depends only on c, δ_n and the value of the register before the first Q cycle. This completes the proof of the lemma. □

Lemma 2.4. $L_3 = \{(a^n b^n)^k \mid n, k \geq 0\} \cup \{(a^n b^{2n})^k \mid n, k \geq 0\} \not\in$ (1DFAM).

Proof.[7] This follows from Lemma 2.2, the observation that $L_3 \cap a^*b^* = L_1$ and the fact that $\mathcal{L}$(1DFAM) is closed under intersection with regular sets. □

Lemma 2.5. *The deterministic context free language* $L_4 = \{x \# x^r \mid x \in \{0,1\}^*\} \not\in \mathcal{L}$(1DFAM).[8]

Proof. Let M be a 1DFAM recognizing L_4. Let s be the number of states of M, r the number of distinct multipliers, $|\Gamma|$, and $n = |x|$. From the proof of Lemma 2.1 it follows that there is a constant, c, such that the number of moves made by M by the time its read head reaches the $\#$ is bounded by cn. Consequently, the number of distinct configurations (i.e., state and register value) of M when its read head is positioned on the $\#$ is bounded by $s(cn)^r$. However for every n, there are 2^n different strings x. Since for every constant, c, there exists an n_0 such that $2^n \geq s(cn)^r$, $n \geq n_0$, it follows that for some n, M has the same configuration while on the $\#$ for two different x, $x_1 \neq x_2$ and $|x_1| = |x_2| = n$. Since M accepts $x_1 \# x_1^r$ and $x_2 \# x_2^r$ and has the same configuration while on the $\#$ for both it follows that M also accepts $x_1 \# x_2^r$ and $x_2 \# x_1^r$. Neither of these is a member of L_4 as $x_1 \neq x_2$. Consequently $T(M) \neq L_4$. □

Lemma 2.6. $L = \{a^i b^j c^k \mid j \neq k \text{ or } i < j\} \not\in \mathcal{L}$(1DFAM).

Proof The proof follows the pattern of Lemmas 2.2 and 2.3 and may be found in [10]. □

We conclude this subsection by showing that all 1DFAM recognizable languages over a one letter alphabet are regular.

Theorem 2.2. *If* $L \in \mathcal{L}$(1DFAM) *and* $|\Sigma| = 1$ *then* L *is regular.*

Proof. We just outline the proof. Given a 1DFAM, M_1, recognizing L, we construct a nondeterministic finite automaton, M_2, such that $T(M_2) = T(M_1)$. Since M_1 is deterministic, there is a constant c such that M_1 can make at most c multiplications on its register before the register value either becomes equal to 1, or if more than c multiplications are made then the register cannot become 1 except on the endmarker \$. This follows from the observation that between two consecutive times when the register value becomes 1, M_1 effectively multiplies the register by $k_0\, p^i\, k_j$

[7] We are grateful to an anonymous referee for providing this proof which shortens our proof.

[8] x^r is the reverse of x.

(k_0 and p depend only on the state q_p of M when its register was last 1 and on Γ; k_j depends only on the state of M_1 when its register was last 1, Γ and the position in the state repetition cycle M_1 is in, $0 < j \leq l$, where l is the length of the state cycle of M_1 for q_p). For $i \geq c_p$ (c_p depends only on q_p, p and the k_j's), $k_0 p^i$ is such that multiplication by any of the k_j's cannot result in the register having a value 1. Finally, the behavior of M_1 on the \$ depends only on the state in which it reaches the \$ and whether or not its register has the value 1. M_2 begins by guessing the state in which M_1 will reach the \$ and also the state in which it will leave the multiplicative cycle on \$. Then it behaves as M_1, performing the multiplications in its finite control (actually, it just keeps track of the exponents in the prime decomposition of the numerator and denominator).

M_1 on the \$ multiplies its register by $k_0'(p')^i k_j'$ (k_0', p', k_j' depending on the state in which M_1 reaches the \$ and the state in which it halts). If M_1 makes more than c_p cycles, then M_2 knows that the register can never become 1 except on the \$. So, M_1 now multiplies its register by $k_0' k_j'$. Then, while moving to the right and simulating M_1, M_2 also multiplies by p' as needed, to keep the prime decomposition bounded. If while doing this, the exponent of any of the primes in the decomposition of the register exceeds a bound dependent only on q_p, p, the k_j's and p', then M_1 on the \$ cannot leave the multiplicative cycle on the \$ in the state guessed by M_2. So M_2 aborts the computation. Clearly, $x \in T(M_2)$ iff $x \in T(M)$. □

2.3. Closure properties of 1FAM's

Theorem 2.3 summarises our results for 1DFAM while Theorem 2.4 does this for 1NFAM.

Theorem 2.3. *The class of languages recognizable by 1DFAM's is not closed under the following operations:*[9]

(a) *union,*
(b) *concatenation,*
(c) *intersection,*
(d) *left quotient with a regular set,*
(e) *ε-free homomorphism,*
(f) *Kleene closure (i.e., $*$),*
(g) *complementation.*

[9] Let X and Y be sets of strings. The *concatenation* of X and Y is the set $XY = \{xy \mid x \text{ in } X, y \text{ in } Y\}$. The *reverse* of X is the set $X^R = \{x^r \mid x \text{ in } X\}$. The *left-quotient* of X with respect to Y is the set $Y \backslash X = \{x \mid yx \text{ in } X \text{ for some } y \text{ in } Y\}$. The *Kleene closure* of X is the set $X^* = \bigcup_{k \geq 0} X^k$, where X^k is defined by $X^0 = \{\varepsilon\}$ and $X^{k+1} = X^k X$ for all $k \geq 0$. Let Σ and Δ be finite sets of symbols. A homomorphism h from Σ^* into Δ^* is any mapping from Σ^* into Δ^* such that $L(\varepsilon) = \varepsilon$ and $h(a_1 \cdots a_k) = h(a_1) \cdots h(a_k)$ for all $k \geq 1$ and a_i in Σ. h is *ε-free* if $h(x) = \varepsilon$ implies $x = \varepsilon$.

Proof. (a) Let $A = \{a^nb^n \mid n \geq 0\}$ and $B = \{a^nb^{2n} \mid n \geq 0\} \in \mathscr{L}(\text{1DFAM})$, then $A \cup B = L_1$ of Lemma 2.2 and $L_1 \notin \mathscr{L}(\text{1DFAM})$.

(b) For the languages A, B of (a) above, we have $AB \cap \{a^*b^*\} = L_1$. Note that $\mathscr{L}(\text{1DFAM})$ is closed under intersection with regular sets.

(c) $A = b^+(\{a^nb^n \mid n \geq 1\})^*$ and $B = (\{b^na^n \mid n \geq 1\})^*b^+$ are in $\mathscr{L}(\text{1DFAM})$. But $A \cap B = \{b^n(a^nb^n)^k \mid n, k \geq 1\}$ is L_2 of Lemma 2.3.

(d) $A = \{a^nb^n \mid n \geq 0\} \cup \{ca^nb^{2n} \mid n \geq 0\} \in \mathscr{L}(\text{1DFAM})$. Consider the regular set $R = \{c, \varepsilon\}$. Then $R \setminus A \notin \mathscr{L}(\text{1DFAM})$ since $(R \setminus A) \cap \{a^ib^j \mid i, j \geq 0\}$ is L_1 of Lemma 2.2.

(e) Just consider $A = \{a^nb^n \mid n \geq 0\} \cup \{ca^{n-1}b^{2n} \mid n \geq 0\}$ and the homomorphism h defined by $h(a) = h(c) = a$ and $h(b) = b$.

(f) $A = \{a^nb^n \mid n \geq 0\} \cup \{ca^nb^{2n} \mid n \geq 0\} \in \mathscr{L}(\text{1DFAM})$. If $A^* \in \mathscr{L}(\text{1DFAM})$ then $B = A^* \cap \{ca^ib^j \mid i, j \geq 0\} = \{ca^nb^n \mid n \geq 0\} \cup \{ca^nb^{2n} \mid n \geq 0\} \in \mathscr{L}(\text{1DFAM})$. However, $B \notin \mathscr{L}(\text{1DFAM})$ since from a 1DFAM, M, recognizing B one easily obtains a 1DFAM, M', recognizing $L_1 = \{a^nb^n \mid n \geq 0\} \cup \{a^nb^{2n} \mid n \geq 0\}$. ($M'$ on the left endmarker ¢ simulates the behavior of M on ¢c and then behaves exactly as M.)

(g) $L = \{a^nb^mc^m \mid n \geq m \geq 0\} \in \mathscr{L}(\text{1DFAM})$ but $\bar{L} \cap \{a^ib^jc^k \mid i, j, k \geq 0\} \notin \mathscr{L}(\text{1DFAM})$ (Lemma 2.6). □

Theorem 2.4. *$\mathscr{L}(\text{1NFAM})$ is a full AFL closed under reversal.*[10]

Proof. The constructions are fairly straightforward and so are omitted. □

Conjectures. *$\mathscr{L}(\text{1DFAM})$ is not closed under reversal. $\mathscr{L}(\text{1NFAM})$ is not closed under intersection and complementation.*

3. FAM without equality

We now turn our attention to FAM's that are not able to check their register for equality to 1. As we shall see, these machines do not have the same closure properties as 1FAM's. In Section 3.1 we obtain the closure properties for 1FAMW. In Section 3.2 we study the relationship between bounded semilinear languages and bounded 1FAMW languages.

3.1. Closure properties of 1FAMW's

Observation 3.1. *For any 1DFAMW, M, if r_i is the value of the register when M*

[10] A nonempty family of sets (or strings) is a full AFL if it is closed under the operations of union, concatenation, Kleene closure, homomorphism, inverse homomorphism and intersection with regular sets.

moves its read head to the right endmarker, \$, and r_0 its value when M halts, then r_i/r_0 can take on only finitely many distinct values.

Proof. This follows from the fact that M is deterministic, has only a finite number of distinct states and a finite number of distinct multipliers. $\square$

Lemma 3.1. (a) $L_1 = \{a^nb^n \mid n \geq 1\} \cup \{a^nb^{2n}c \mid n \geq 1\} \notin \mathscr{L}(\text{1DFAMW})$.
(b) $L_2 = \{a^nb^n \mid n \geq 0\}^* \notin \mathscr{L}(\text{1NFAMW})$.

Proof. (a) Let M be a 1DFAMW accepting L_1. M has a finite number of states, s. So, for infinitely many n's, $(n_1, n_2, \cdots, n_r, \cdots)$, it must be the case that M enters the first b of $x = a^nb^n$ in the same state q'. For each of these n's $> s$, M must repeat some state. Let q_i be the first state that is repeated on the b's. Let δ be the multiplicative factor by which the register changes between two consecutive appearances of the state q_i. Since M is deterministic and its moves are independent of the register value, the cycle length, l, (i.e., the distance on the input tape between two consecutive appearances of q_i) and δ are fixed over the segment of b's. $\delta \neq 1$ as otherwise deletion or addition of l b's to x would result in a string $x' \notin L_1$ but $x' \in T(M)$. Note that l, δ are fixed for all the n's, $(n_1, n_2, \cdots, n_r, \cdots)$ we are considering as l, δ depend only on the state in which M enters the first b. By Observation 3.1, since M accepts all the infinitely many inputs $x_i = a^{n_i}b^{n_i}$, it follows that for some infinite subset of these n's, $(\bar{n}_1, \bar{n}_2, \cdots, \bar{n}_m, \cdots)$, M's register has the same value, k_i, when M moves right from the last b of $x_i = a^{\bar{n}_i}b^{\bar{n}_i}$. Since M is deterministic, has only a finite number of multipliers and $\delta \neq 1$, it follows that for infinitely many n's, M arrives at the c of $y_i = a^{\bar{n}_i}b^{2\bar{n}_i}c$ with a different register value (i.e., when M is reading the c, its register could have any one of infinitely many distinct values depending on $\bar{n}_i$). This in turn implies that M can reach the \$ with infinitely many different register values. Observation 3.1 then implies that M cannot accept all these y_i's. Consequently $T(M) \neq L_1$. Note that $L_1 \in \mathscr{L}(\text{1DFAM})$.

(b) Assume there is a 1NFAMW, M, such that $T(M) = L_2$. Let s be the number of states of M. The string $x = d_1d_2 \cdots d_{n_1}$, $d_i = a_{i1}a_{i2} \cdots a_{in_1}b^{n_1}$ $(a_{ij} = a, 1 \leq i, j \leq n_1)$ and $n_1 > s$, is a member of L_2. Consequently, M has a computation on x that results in an accepting configuration. For each sumbol a_{ij} $(a_{ij} = a)$ let q_{ij} be the state M is in when its read head moves onto a_{ij} in this accepting computation on x. Since $n_1 > s$, it must be the case that for each d_i, M is in the same state at least twice on the d_i's. i.e., for every i there exist $j, l, 0 < j < l \leq n_1$ such that $q_{ij} = q_{il}$. For each d_i, let $\bar{q}_i$ be one such state being repeated and let a_{ij_i} and a_{ik_i} be two a's in d_i where the entering state is $\bar{q}_i$ $(0 < j_i < k_i \leq n_1)$. There are $n_1 > s$ segments of d's. So, for some m and l, $0 < m < l \leq n_1$, $\bar{q}_m = \bar{q}_l$. Since the moves of M are independent of the value of its register, and multiplication is commutative, M will have an accepting computation on the string y obtained from x by moving $(k_m - j_m)$ a's from d_m to the segment of a's in d_l. But, $y \notin L_2$. Hence, $L_2 \notin \mathscr{L}(\text{1NFAMW})$. $\square$

Theorem 3.1. $\mathscr{L}$(1DFAMW) *is closed under intersection but not closed under the following operations*:

(a) *union*,
(b) *concatenation*,
(c) *Kleene closure*,
(d) *ε-free homomorphism*,
(e) *left quotient with a regular set*,
(f) *complementation*,
(g) *reversal.*

Proof. For (a)–(e), the proofs of the corresponding results for $\mathscr{L}$(1DFAM) (Theorem 2.3) also apply here. Nonclosure under complementation follows from (a), closure under intersection (shown below) and De Morgan's law. That $\mathscr{L}$(1DFAMW) is not closed under reversal follows from the fact that $L_1 = \{b^n a^n \mid n \geq 1\} \cup \{cb^{2n}a^n \mid n \geq 1\}$ is in $\mathscr{L}$(1DFAMW) while its reversal is not (Lemma 3.1 (a)). The proof for closure under intersection is by construction. If L_1 and $L_2 \in \mathscr{L}$(1DFAMW) then there exist 1DFAMW's M_1 and M_2 such that $T(M_1) = L_1$ and $T(M_2) = L_2$. We construct a 1DFAMW M_3 such that $T(M_3) = L_1 \cap L_2$. The details of the proof are omitted and may be found in [10]. First, 1DFAMW's M_1' and M_2' are obtained such that the primes in the prime decompositions of the factors of M_1' and M_2' are disjoint and $T(M_1') = T(M_1)$, $T(M_2') = T(M_2)$. Next, an M_3 is constructed from M_1' and M_2'. It is essentially M_1' and M_2' operating in parallel. Whenever M_1' and M_2' multiply their register by r_1 and r_2, M_3 multiplies its register by $r_1 r_2$. Since the primes in the multipliers of M_1' and M_2' are disjoint, the register value for M_3 is 1 iff the registers of both M_1' and M_2' have value 1. □

Theorem 3.2. *The class of languages recognized by 1NFAMW's is closed under the operations of union, reversal, homomorphism, left-quotient with a regular set, intersection, and concatenation. The class is not closed under Kleene closure and complementation.*

Proof. Closure under union, reversal, homomorphism and left-quotient with a regular set are easily verified. The proof for intersection is similar to the one outlined in Theorem 3.1. The proof for concatenation is also obvious if we note that when we are dealing with two 1NFAMW's M_1 and M_2, we may assume that the primes occurring in the prime decompositions of the multipliers of M_1 and M_2 are distinct (see the proof of Theorem 3.1). The language $L_2 = \{a^n b^n \mid n \geq 0\}^* \notin \mathscr{L}$(1NFAMW). Nonclosure under Kleene closure and complementation now follow from the fact that $\{a^n b^n \mid n \geq 0\}$ and the complement of L_2 are in $\mathscr{L}$(1NFAMW).

3.2. Relationship to bounded semilinear languages

We now look at the class of bounded languages accepted by 1NFAMW's. We show that this class coincides with the class of semilinear bounded languages [6].

A set $L \subseteq \Sigma^*$ is bounded if there exist nonnull strings $w_1, \cdots, w_n$ in Σ^* such that $L \subseteq w_1^* \cdots w_n^*$.[11]

Let $\mathbf{N}$ be the set of natural numbers and $\mathbf{N}^n$ the set of n-tuples of natural numbers. A subset Q of $\mathbf{N}^n$ is called a *linear set* if there exist $v_0, v_1, \cdots, v_m$ in $\mathbf{N}^n$ such that $Q = \{v_0 + k_1 v_1 + \cdots + k_m v_m \mid k_i \text{ in } \mathbf{N}\}$. $v_0, \cdots v_m$ are the *generators* of Q; v_0 is called the *constant* and $v_1, \cdots, v_m$ the *periods.* Any finite union of linear sets is called a *semilinear set.* If $L \subseteq w_1^* \cdots w_n^*$, define the mapping $f_{\langle w_1, \cdots, w_n \rangle}$ of L as follows:

$$f_{\langle w_1, \cdots, w_n \rangle}(L) = \{(i_1, \cdots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \text{ in } L\}.$$

Thus, $f_{\langle w_1, \cdots, w_n \rangle}(L) \subseteq \mathbf{N}^n$. If $f_{\langle w_1, \cdots, w_n \rangle}(L)$ is a semilinear set, then L is called a bounded *semilinear language.* In [6] it is shown that semilinear sets with certain properties characterize exactly the bounded context-free languages.

In [9, 15, 16] it is shown that bounded semilinear languages are exactly those bounded languages accepted by one-way nondeterministic multihead finite automata. We now show a similar result for 1NFAMW's.

Theorem 3.3. *Let $L \subseteq w_1^* \cdots w_n^*$ be a bounded semilinear language. Then L can be accepted by a 1NFAMW in linear time.*

Proof. Since L is a bounded semilinear language, $f_{\langle w_1, \cdots, w_n \rangle}(L)$ is a semilinear set, Q. Now the class of languages accepted by 1NFAMW's operating in linear time is clearly closed under union. Hence it is sufficient to show that L is accepted by a 1NFAMW in linear time for Q, a linear set. Let Q be generated by $v_0, v_1, \cdots, v_m$, where $v_0 = (b_1, \cdots, b_n)$ and $v_i = (v_{i1}, \cdots, v_{in})$ for $1 \leqslant i \leqslant m$. We may assume that $v_1, \cdots, v_m$ are distinct from $(0, \cdots, 0)$. Then

$$L = \{w_1^{b_1}(w_1^{v_{11}})^{k_1} \cdots (w_1^{v_{m1}})^{k_m} w_2^{b_2}(w_2^{v_{12}})^{k_1} \cdots (w_2^{v_{m2}})^{k_m} \cdots w_n^{b_n}(w_n^{v_{1n}})^{k_1} \cdots (w_n^{v_{mn}})^{k_m} \mid k_i \geqslant 0 \text{ for } 1 \leqslant i \leqslant n\}.$$

Let c_i, a_{ij} $(1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m)$ be distinct symbols and

$$R = \{c_1 a_{11}^{k_1} a_{12}^{k_2} \cdots a_{1m}^{k_m} c_2 a_{21}^{k_1} a_{22}^{k_2} \cdots a_{2m}^{k_m} \cdots c_n a_{n1}^{k_1} a_{n2}^{k_2} \cdots a_{nm}^{k_m} \mid k_i \geqslant 0, 1 \leqslant i \leqslant m\}.$$

For each $2 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$, let p_{ij} be a distinct prime number. R is accepted by a 1DFAMW (in real-time) operating as follows: When M reads each a_{1j}, it multiplies by $p_{2j} \cdots p_{nj}$, when M reads a_{ij}, $i > 1$, it multiplies by $1/p_{ij}$. If the number of a_{ij}'s is the same as the number of a_{1j}'s, the p_{ij}'s will cancel out by the time M sees the \$. Now define a homomorphism h by: For $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$, let $h(c_i) = w_i^{b_i}$ and

[11] For strings $w_1, \cdots, w_n$ in Σ^*, we write $w_1^* \cdots w_n^*$ to denote the set $\{w_1^{i_1} \cdots w_n^{i_n} \mid i_1, \cdots, i_n \geqslant 0\}$.

$h(a_{ij}) = w_i^{v_{ij}}$. Then, $h(R) = L$. Since $v_1, \cdots, v_m$ are non-zero n-tuples, there exists $k \geq 1$ such that $k\,|h(x)| \geq |x|$ for all $x \neq c_1c_2\cdots c_n$ in R. Using this condition, we can easily construct a 1NFAMW M' accepting $h(T(M))$ and which operates in linear time. □

Before we can prove the converse of Theorem 3.3 we need a lemma connecting 1NFAMW with a special class of one-way multihead finite automata.

One-way multihead finite automata have been studied in several places in the literature (see, e.g., [7, 14, 15]). Here we are interested in a special class which is related to 1NFAMW and 1DFAMW.

A *one-way nondeterministic k-head finite automaton* (k-NFA) is a device $M = \langle k, K, \Sigma, \delta, q_0, ¢, \$, F\rangle$, where k is the number of input heads, k, Σ, and F are finite sets of states, input symbols, and accepting states, q_0 in K is the initial state, ¢ and \$ are the left and right endmarkers for the inputs, and δ is a mapping from $K \times (\Sigma \cup \{¢, \$\})^k$ into the set of all subsets of $K \times \{0, 1\}^k$. The significance of $(p, d_1, \cdots, d_k)$ in $\delta(q, a_1, \cdots, a_k)$ is that if M is in state q with head i reading a_i on the input $(1 \leq i \leq k)$, M may change state to p and move head h_i to the right if $d_i = 1$ and does not move it if $d_i = 0$. A string x in Σ^* is *accepted* if M when started in state q_0 with all input heads on the left endmarker of $¢x\$$ eventually enters an accepting state with all heads on the right endmarker. We denote by $T(M)$ the set of all strings accepted by M. A *multihead NFA* is a k-NFA for some k.

We shall only be concerned with a special class of one-way k-head finite automata: A *simple k-NFA* is a k-NFA $M = \langle k, K, \Sigma, \delta, q_0, ¢, \$, F\rangle$ with the restriction that for each $2 \leq i \leq k$, q in K, $a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_k$ in $\Sigma \cup \{¢,\$\}$,

$$\delta(q, a_1, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_k) = \delta(q, a_1, \cdots, a_{i-1}, b_i, a_{i+1}, \cdots, a_k)$$

for all a_i, b_i in Σ. Thus, a simple k-NFA is a k-NFA in which all heads except the first head cannot distinguish symbols in Σ. We refer to the first head as the *input head* and the other heads as *counting heads*. A *simple multihead NFA* is a simple k-NFA for some k.

A k-NFA (simple k-NFA) $M = \langle k, K, \Sigma, \delta, q_0, ¢, \$, F\rangle$ is deterministic if $|\delta(q, a_i, \cdots, a_k)| \leq 1$ for all q in K, $a_1, \cdots, a_k$ in $\Sigma \cup \{¢, \$\}$. We write *k-DFA* and *simple k-DFA* for the deterministic cases.

The following lemma connects 1NFAMW (1DFAMW) to simple multihead NFA(DFA).

Lemma 3.2. *Let $L = T(M)$ for some 1NFAMW (1DFAMW) $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F\rangle$. Let c be some positive integer and let $L(cn) = \{x \mid x$ in L, M accepts x in $\leq c\,|¢x\$|$ steps$\}$. Then $L(cn)$ is accepted by a simple multihead NFA (DFA), M'.*

Proof. Let $P = \{p_1, \cdots, p_k\}$ be the set of all primes appearing in the prime decompositions of the numerators and denominators of multipliers in Γ and

$e = \max\{|e_i| \mid p_1^{e_1} \cdots p_k^{e_k}$ is in $\Gamma\}$. For each p_i, we associate two counting heads N_i and D_i. N_i and D_i will keep track of "counts" initially set to zero. During the simulation the counts of N_i and D_i are updated as follows: Each time M multiplies the register by m, the count of N_i or D_i is increased by m_i depending on whether the m_i occurrences of the prime p_i is in the numerator or denominator of m. Actually, N_i (or D_i) is only moved one square right for every increase of ce units in the count. Thus, M has $2k$ counting heads associated with the primes in P and an input head to simulate the input head of M. If during the simulation, M enters an accepting state with the input head on \$, M' moves all its counting heads to the right endmarker and checks that heads N_i and D_i arrive at the right endmarker at the same time (for $1 \leq i \leq k$). (This corresponds to the register having value 1). If such is the case, M' accepts the input. It is clear that $T(M') = L(cn)$. □

The next result is taken from [9, 15, 16].

Lemma 3.3. *Let $L \subseteq w_1^* \cdots w_n^*$ be recognizable by some multihead NFA, M. Then $f_{\langle w_1,\ldots,w_n\rangle}(L)$ is a semilinear set effectively computable from M.*

Theorem 3.4. *Let $L \subseteq w_1^* \cdots w_n^*$ and $L = T(M)$ for some 1NFAMW, M. Then L is a bounded semilinear language. Moreover, the corresponding semilinear set Q is effectively computable from M.*

Proof. Let d be a new symbol and let $L' = \{xd^k \mid k \geq 1, x \text{ in } L\}$. We construct a 1NFAMW M' accepting L' which operates as follows: Given $¢xd^k\$$, M' simulates M on x treating the first d like \$. When M accepts x, M' moves its input head to \$ and enters an accepting state. It is obvious that for each input x, there exists an integer i_k such that M' accepts xd^{i_x} in $\leq 2|¢xd^{i_x}\$|$ steps. By Lemma 3.2, $L'(2n)$ is accepted by some simple multihead NFA. It follows from Lemma 3.3, that $f_{\langle w_1,\ldots,w_n,d\rangle}(L'(2n))$ is a semilinear set, say Q. By deleting the $(n+1)$st coordinates from the generators of the linear sets making up Q, we obtain the generators of the semilinear set $f_{\langle w_1,\ldots,w_n\rangle}(L)$. □

Corollary 3.1. *All 1NFAMW recognizable languages over a one letter alphabet, i.e., $|\Sigma| = 1$, are regular.*

Proof. This follows from the above theorem and the fact that all bounded semilinear languages over one letter alphabets are regular. □

4. Inclusion relations and decidability results

It is interesting to note the relationships among the different classes of FAM's. The next theorem lists some of these relationships.

Theorem 4.1. *The following relationships hold among the classes of languages accepted by the various FAM's.*

(a) $\mathscr{L}(\text{1DFAM}) \subsetneq \mathscr{L}(\text{1NFAM})$,
(b) $\mathscr{L}(\text{1DFAM}) \subsetneq \mathscr{L}(\text{2DFAM})$,
(c) $\mathscr{L}(\text{1DFAMW}) \subsetneq \mathscr{L}(\text{1NFAMW})$,
(d) $\mathscr{L}(\text{1DFAMW}) \subsetneq \mathscr{L}(\text{2DFAMW})$,
(e) $\mathscr{L}(\text{1NFAMW}) \subsetneq \mathscr{L}(\text{2NFAMW})$,
(f) $\mathscr{L}(\text{1DFAMW}) \subsetneq \mathscr{L}(\text{1DFAM})$,
(g) $\mathscr{L}(\text{1NFAMW}) \subsetneq \mathscr{L}(\text{1NFAM})$.

Proof. (a) and (c) follow from the fact that $L = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0$ $\in \mathscr{L}(\text{1NFAMW})$ and $L \notin \mathscr{L}(\text{1DFAM})$ (see Lemma 2.2). (b) follows from Lemma 2.3 and the observation that $L = \{b^n(a^n b^n)^k \mid k \geq 1\} \in \mathscr{L}(\text{2DFAM})$.

For (d) consider the language $L_1 = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} c \mid n \geq 0\}$ of Lemma 3.1. $L_1 \notin \mathscr{L}(\text{1DFAMW})$ but clearly $L_1 \in \mathscr{L}(\text{2DFAMW})$ (a 2DFAMW accepting L_1 works by first determining the presence or absence of the "c", returning to the left endmarker "¢" and then proceeding to recognize $a^n b^n$ or $a^n b^{2n}$ depending on whether or not there was a "c" on the right end).

(e) Consider the language $L = \{a^n b^{kn} \mid n \geq 2, k \geq 1\}$. Clearly, $L \in \mathscr{L}(\text{2NFAMW})$. (The 2NFAMW nondeterministically makes k passes over the a segment, multiplying its register by 2 each time an a is encountered. It then scans the b's multiplying its register by $\frac{1}{2}$.) Suppose $L \in \mathscr{L}(\text{1NFAMW})$. Then by Theorem 3.2, $h(L)$ is also accepted by a 1NFAMW, where h is the homomorphism $h(a) = h(b) = a$. Then by Corollary 3.1, $h(L)$ is regular. But $h(L) = \{a^{kn} \mid k \geq 2, n \geq 2\}$ is the set of all strings a^l for which l is not prime, which is not regular. We conclude that $L \notin \mathscr{L}(\text{1NFAMW})$.

(f) L_1 of Lemma 3.1 is not 1DFAMW recognizable but $L_1 \in \mathscr{L}(\text{1DFAM})$.

(g) L_2 of Lemma 3.1 is not 1NFAMW recognizable but $L_2 \in \mathscr{L}(\text{1DFAM})$. □

Observation 4.1. *There exist languages L_1 and L_2 such that $L_1 \in \mathscr{L}(\text{1DFAM})$ but $L_1 \notin \mathscr{L}(\text{1NFAMW})$ and $L_2 \in \mathscr{L}(\text{1NFAMW})$ but $L_2 \notin \mathscr{L}(\text{1DFAM})$.*

Proof. $L_1 = \{a^n b^n \mid n \geq 0\}^*$, $L_2 = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$. □

We now investigate some decidability questions concerning FAM's. We shall reduce their decision problems to the solvability of the emptiness and containment problems for simple multihead NFA's and simple multihead DFA's, respectively. In [7], it is shown that every set accepted by a multihead NFA over a one-letter input alphabet is regular. A close look at the proof shows that, in fact, a finite automaton accepting the set can be effectively constructed. Thus, we may state the following lemma.

Lemma 4.1. *Let M be a multihead NFA over a one-letter input alphabet. Then we can effectively find a finite automaton M' accepting the set $T(M)$.*

Next, we observe that the class of languages recognized by simple multihead DFA's is closed under the operations of intersection, union, and complementation.

Lemma 4.2. *If M_1 and M_2 are simple multihead DFA's with input alphabet Σ, then we can effectively find simple multihead DFA's M_3, M_4 and M_5 such that*

(a) $T(M_3) = T(M_1) \cap T(M_1)$,

(b) $T(M_4) = T(M_1) \cup T(M_2)$,

(c) $T(M_5) = \Sigma^* - T(M_1)$.

Thus the class of languages recognized by simple multihead DFA's is a Boolean algebra.

Proof. The construction of M_3, M_4 and M_5 uses standard techniques and is therefore omitted. Formal construction can be found in [10]. □

We can now established the solvability of the emptiness and containment problems for simple multihead NFA's and simple multihead DFA's, respectively.

Theorem 4.2. *The following problems are solvable (i.e., decision algorithms exist):*

(a) *Emptiness problem: Given an arbitrary simple multihead NFA M, determine whether or not $T(M) = \emptyset$.*

(b) *Containment problem: Given two arbitrary simple multihead DFA's M_1 and M_2, determine whether or not $T(M_1) \subseteq T(M_2)$.*

(c) *Equivalence problem: Given two arbitrary simple multihead DFA's M_1 and M_2, determine whether or not $T(M_1) = T(M_2)$.*

Proof. (a) Given $M = \langle k, K, \Sigma, \delta, q_0, ¢, \$, F\rangle$, construct a simple multihead NFA M' accepting the set $h(M)$, where h is a homomorphism defined by $h(a) = \#$ for each a in Σ, $\#$ is a new symbol not in Σ. The construction of M' is obvious. Then $T(M) = \emptyset$ if and only if $T(M') = \emptyset$. By Lemma 4.1, we can construct a finite automaton M'' accepting $T(M')$. The result now follows from the solvability of the emptiness problem for finite automata [13].

(b) Given simple multihead DFA's M_1 and M_2, we construct (using Lemma 4.2) a simple multihead DFA M_3 such that $T(M_3) = \Sigma^* - T(M_2)$. Then $T(M_1) \subseteq T(M_2)$ if and only if $T(M_1) \cap T(M_3) = \emptyset$. By Lemma 4.2, we can construct a simple multihead DFA M_4 accepting $T(M_1) \cap T(M_3)$. Then $T(M_1) \subseteq T(M_2)$ if and only if $T(M_4) = \emptyset$ which is solvable by (a).

(c) This follows from (b). □

We now mention a few solvable problems concerning FAM's.

Theorem 4.3. *The emptiness problem for 1NFAMW is solvable.*

Proof. Given a 1NFAMW $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F\rangle$, define the set $L_M = \{xd^k \mid x$ in $T(M), k \geq 1\}$ (d is a new symbol not in Σ). Clearly we can construct a 1NFAMW M_1 recognizing L_M (see the proof of Theorem 3.4). By Lemma 3.2 we can construct a simple multihead NFA M_2 recognizing $L_M(2n) = \{x \mid x$ in L_M, M_1 accepts x in $\leq 2 \mid ¢x\$ \mid$ steps$\}$. Now $T(M_2) = \emptyset$ if and only if $T(M_1) = \emptyset$ which is true if and only if $T(M) = \emptyset$. The result follows from Theorem 4.2. □

Lemma 2.1 and Corollary 2.1 show that 1DFAM and 2DFAM operate in time $O(n)$ and $O(n^3)$, respectively. It is obvious that these results apply to 1DFAMW and 2DFAMW also. Hence the membership problem (i.e., deciding whether or not an arbitrary input to an arbitrary machine is accepted) is solvable. At present, we do not have an algorithm for the membership problem for 1NFAM's and 2NFAM's. The following result shows the problem to be solvable for 2NFAMW's.

Corollary 4.1. *There is an algorithm to decide given a 2NFAMW* $M = \langle K, \Sigma, \Gamma, \delta, q_0, ¢, \$, F\rangle$ *and* x *in* Σ^* *whether or not* x *is in* $T(M)$.

Proof. Given M and x, we construct a 1NFAMW M_x which when given an input $¢y\$$, disregards the input and simulates in its finite control the computation of M on $¢x\$$. M_x accepts y if and only if M accepts x. It follows that x is in $T(M)$ if and only if $T(M_x) \neq \emptyset$ which is solvable by Theorem 4.2. □

From Lemma 2.1 and Lemma 3.2, we know that we can effectively construct a simple multihead DFA accepting the language accepted by a given 1DFAMW. The next result then follows Theorem 4.1.

Theorem 4.4. *The emptiness, containment, and equivalence problems for 1DFAMW's are solvable.*

We now prove a result showing the unsolvability of the universe problem (i.e., deciding whether or not $T(M) = \Sigma^*$) for 1NFAMW's, a contrasting result from that of Theorem 4.4.

Theorem 4.5. *It is unsolvable to decide given an arbitrary 1NFAMW (with input alphabet* Σ*) whether or not it accepts* Σ^*.

Proof. Given a single-tape Turing machine (see [8]) M, define a string x_M representing a valid computation of M on an initially blank tape (if it exists) as a sequence $x_M = \alpha_0 \# \alpha_1 \# \cdots \# \alpha_k$, where α_0 is the initial configuration of M, α_{i+1} is the configuration resulting from α_i after one move of M, and α_k is a halting

configuration. Let the tape alphabet of M be Δ and its state set be K. Then x_M, if it exists, is in $(\Delta \cup K \cup \{\#\})^*$.
Now define the set

$$L_M = \begin{cases} (\Delta \cup K \cup \{\#\})^* - \{x_M\} & \text{if } x_M \text{ exists,} \\ (\Delta \cup K \cup \{\#\})^* & \text{otherwise.} \end{cases}$$

Then $L_M = \Sigma^*$ (where $\Sigma = \Delta \cup K \cup \{\#\}$) if and only if x_M does not halt on an initially blank tape. Now can easily construct a 1NFAMW to accept L_M. (The construction is left as an exercise to the reader.) The theorem now follows from the unsolvability of the halting problem for Turing machines on blank tape [18]. (Note that by an appropriate coding, Σ can be reduced to a 2-letter alphabet.) □

We conclude this section with two applications of FAM's to decision problems concerning vector addition systems and integer programming problems.

A *vector addition system* [11] is a pair $S = (v_0, V)$, where v_0 is an n-dimensional vector of nonnegative integers, and V is a finite set of n-dimensional integer vectors. The *reachability set* $R(S)$ is the set of all vectors of the form $v_0 + v_{i_1} + \cdots + v_{i_k}$ such that

(1) v_{i_j} is in V for $1 \leq j \leq k$, and

(2) $v_0 + v_{i_1} + \cdots + v_{i_j} \geq 0$ for $1 \leq j \leq k$.

It is not known whether or not there is an algorithm to decide given an n-dimensional vector addition system S and a vector v whether v is in the reachability set $R(S)$ (see [11]). However, it was pointed out in [11], that the equivalence problem for reachability sets is unsolvable.

Suppose we relax the definition of the reachability set by deleting requirement (2) above. Call this new set *relaxed reachability set*, $T(S)$. Then we can show that the equivalence problem is decidable.

Theorem 4.6. *The equivalence problem for relaxed reachability sets is decidable.*

Proof. Let $S = (v_0, V)$ be an n-dimensional vector addition system, $V = v_1, \cdots, v_m$. Let $v_0 = (e_{01}, \cdots, e_{0n})$ and $v_i = (e_{il}, \cdots, e_{in})$, $1 \leq i \leq m$. For $0 \leq i \leq m$, let $t_i = p_1^{e_{i1}} \cdots p_n^{e_{in}}$, where $p_1, \cdots, p_n$ are the first n primes. Let $a_1, \cdots, a_n$ be distinct symbols, and define the language

$$L_S = \{c_1 a_1^{i_1} \cdots c_n a_n^{i_n} \mid c_i \text{ in } \{+, -\}, (c_1 i_1, \cdots, c_n i_n) \text{ in } T(S)\}.$$

L_S can be accepted by a 1NFAMW M operating as follows: For each a_j $(1 \leq j \leq n)$, M multiplies the register by p_j if c_j is $-$ and by $1/p_j$ if c_j is $+$. When M reaches \$, it nondeterministically multiples the register by the product $t_0 t_1^{k_1} \cdots t_m^{k_m}$ for some $k_1 \geq 0, \cdots, k_m \geq 0$, and accepts if the register becomes 1. Clearly, $T(M) = L_S$. Now $T(M) \subseteq +^* -^* a_1^* \cdots +^* -^* a_n^*$ is bounded. By Theorem 3.4, the set

$$Q_S = \{(j_1, l_1, i_1, \cdots, j_n, l_n, i_n) \mid +^{j_1} -^{l_1} a_1^{i_1} \cdots +^{j_n} -^{l_n} a_n^{i_n} \text{ in } T(M)\}$$

is a semilinear set effectively computable from M. Now if we are given two vector addition systems S_1 and S_2, we can construct their associated semilinear sets, Q_{S_1} and Q_{S_2}. Then $T(S_1) = T(S_2)$ if and only if $Q_{S_1} = Q_{S_2}$ which is decidable [6].

As another application of the notion of 1NFAMW, consider the system of equations of the form $FX = G$, where F is an $n \times m$ matrix of integers, X is a vector of m variables, and G is a vector of n integers. We shall show that the set of all nonnegative integer solutions X to the system of equations forms an effectively computable semilinear set. This result has been shown earlier in [19]. □

Theorem 4.7. *The set of nonnegative integer solutions to the system of equations, $FX = G$ forms an effectively computable semilinear set.*

Proof. Let F be an $n \times m$ matrix. Let $a_1, \cdots, a_m$ be distinct symbols. Define the language

$$L = \{a_1^{i_1} a_2^{i_2} \cdots a_m^{i_m} \mid (i_1, \cdots, i_m) \text{ is a nonnegative integer solution to equations, } FX = G\}.$$

One can easily construct a 1NFAMW M to accept L. By Theorem 3.4, the set $Q = \{(i_1, \cdots, i_m) \mid a_1^{i_1} \cdots a_m^{i_m} \text{ in } L\}$ is a semilinear set effectively computable from M. □

As a corollary, we obtain the following result which was recently shown in [1].

Corollary 4.2. *It is decidable whether or not the system of equations $FX = G$ has a nonegative integer solution.*

5. Conclusion

In conclusion, we summarize our results and state a few unresolved problems concerning FAM's.

(1) The closure properties of the families of languages recognized by 1NFAM's, 1DFAM's, 1NFAMW's and 1DFAMW's are summarized in Table 1.

(2) The class of languages accepted by 2-way machines properly contains the class of languages accepted by corresponding 1-way machines. Also for 1-way machines, nondeterministic machines are more powerful than deterministic ones.

(3) Some of the interesting questions that remain unanswered are:

(i) Is the class of languages accepted by 1NFAM closed by 1NFAM closed under intersection and complement? We conjecture that it is not. Our counterexample is $L = \{b^n(a^nb^n)^k \mid n \geq 1, k \geq 1\}$ but we have no proof.

(ii) Is every bounded language accepted by 1NFAM (or 1DFAM) semi-linear?

TABLE 1.

operations \ FAM's	1NFAM	1DFAM	1NFAMW	1DFAMW
union	yes	no	yes	no
intersection	?	no	yes	yes
complement	?	no	no	no
concatenation	yes	no	yes	no
reversal	yes	?	yes	no
Kleene closure	yes	no	no	no
(ε-free) homomorphism	yes	no	yes	no
left quotient with regular set	yes	no	yes	no

References

[1] B. Baker, On finding non-negative integer solutions to sets of linear equations, Harvard University, Report (1974).

[2] S. Ginsburg and S. Greibach, Deterministic context-free languages, *Information and Control* **9** (1966) 620–648.

[3] S. Ginsburg and S. Greibach, Abstract families of languages, in: *Studies In Abstract Families of Languages*, AMS Memoir **87** (Am. Math. Soc., Providence, R. I., 1969) 1–32.

[4] S. Ginsburg, S. Greibach and M.A. Harrison, One-way stack automata, *J. ACM* **14** (1967) 389–418.

[5] S. Ginsburg and M. Harrison, On the closure of AFL under reversal, *Information and Control* **17** (1970) 395–409.

[6] S. Ginsburg and E.H. Spanier, Bounded ALGOL-like languages, *Trans. Am. Math. Soc.* **113** (1964) 333–368.

[7] M.A. Harrison and O.H. Ibarra, Multi-tape and multi-head pushdown automata, *Information and Control* **13** (1968) 433–470.

[8] J.E. Hopcroft and J.D. Ullman, *Formal Languages and their Relation to Automata* (Addison–Wesley, Reading, Mass., 1969).

[9] O.H. Ibarra, A note on semilinear sets and bounded-reversal multihead pushdown automata, *Information Processing Lett.* **3** (1974) 25–28.

[10] O. H. Ibarra, S. Sahni and C. E. Kim, Finite automata with multiplication, University of Minnesota, Technical Report 74–7 (May 1974).

[11] R.M. Karp and R.E. Miller, Parallel program schemata, *J. Comput. System Sci.* **3** (1969) 147–195.

[12] M.L. Minsky, Recursive unsolvability of Post's problem of "Tag" and other topics in the theory of Turing machines, *Annals of Math.* **74** (1961) 437–455. See also *Computation: Finite and Infinite Machines* (Prentice–Hall, Englewood Cliffs, N. J., 1967).

[13] M.O. Rabin and D. Scott, Finite automata and their decision problems, *IBM J. Res. Develop.* **3** (1959) 114–125.

[14] A.L. Rosenberg, Nonwriting extensions of finite automata, Ph.D. Thesis, Harvard University (1965).

[15] I.H. Sudborough, Bounded-reversal multi-head finite automata languages, *Information and Control* **25** (1974) 317–328.

[16] I.H. Sudborough, Personal Communication.

[17] S. Tamura and K. Tanaka, Note on analogue memory automata, *Information Sci.* **7** (1974) 74–80.

[18] A.M. Turing, On computable numbers with an application to the Entscheidungs problem, *Proc. London Math. Soc.* **42** (1936) 230–265.

[19] J. Van Leeuwen, A partial solution to the reachability problem for vector addition systems, *Stochastics* (1974) 303–309.