JOURNAL OF COMPUTER AND SYSTEM SCIENCES 19, 256–276 (1979)

Reset Machines*

RONALD V. BOOK

Department of Mathematics, University of California at Santa Barbara, Santa Barbara, California 93106

SHEILA A. GREIBACH

Department of System Science, University of California at Los Angeles, Los Angeles, California 90024

AND

CELIA WRATHALL

Department of Mathematics, University of California at Santa Barbara, Santa Barbara, California 93106

Received September 11, 1978; revised March 18, 1979

A reset tape has one read-write head which moves only left-to-right except that the head can be reset once to the left end and the tape rescanned; a multiple-reset machine has reset tapes as auxiliary storage and a one-way input tape. Linear time is no more powerful than real time for nondeterministic multiple-reset machines and so the family MULTI-RESET of languages accepted in real time by nondeterministic multiple-reset machines is closed under linear erasing. MULTI-RESET is closed under Kleene*. It can be characterized as the smallest family of languages containing the regular sets and closed under intersection and linear-erasing homomorphic duplication or as the smallest intersection-closed semiAFL containing COPY $= \{ww \mid w \text{ in } \{a, b\}^*\}$. A circular tape is read full-sweep from left-to-right only and then reset to the left, any number of times; a nonwriting circular tape cannot be altered after the first sweep. For nondeterministic machines operating in real time, multiple reset tapes, circular tapes or nonwriting circular tapes have the same power. Languages in MULTI-RESET can be accepted in real time by nondeterministic machines using only three reset tapes or using only one reset tape and one nonwriting circular tape.

INTRODUCTION

The computation of a Turing machine that is allowed to make only a bounded number of reversals can be described in terms of "write" and "compare" actions. Describing the computations of multitape reversal-bounded machines in this way shows that "last-in,

* Some of the results presented here were announced at the Fifth International Colloquium on Automata, Languages and Programming, July 1978, Udine, Italy. An extended abstract appears in the Proceedings of that symposium. This research was supported in part by the National Science Foundation under Grants MCS77-11360, DCR74-15091, and MCS78-04725.

0022-0000/79/060256-21$02.00/0

first-out'' comparisons are extremely powerful when the machine is allowed to operate nondeterministically and far less powerful when the deterministic mode is required [1]. In this paper we investigate nondeterministic Turing machine models of two other types of comparison: "first-in, first-out'' and "shift.'' The results characterize a number of classes of languages specified by nondeterministic Turing machines that operate in real time, in linear time, in polynomial time, or without time bound, and that have either a single storage tape or multiple storage tapes of certain types.

The "first-in, first-out'' comparison studied here is modeled by a *reset tape*, a tape with one read-write head such that the tape is one-way infinite (to the right) and the head moves only left-to-right except that the head can be reset once to the left end and the tape rescanned. We first consider nondeterministic acceptors with an input tape that is read in only one direction and a single reset tape as auxiliary storage. The class of languages recognized by such machines, single-reset languages, has properties very similar to those of the class of linear context-free languages. While the class of linear context-free languages is generated by the language $PAL = \{ww^R \mid w \in \{a, b\}^*\}$, the class of single-reset languages is generated by the language $COPY = \{ww \mid w \in \{a, b\}^*\}$.

A *multiple-reset machine* is a multitape machine with reset tapes as auxiliary storage. The class of arbitrary nondeterministic multiple-reset machines accept exactly the class of recursively enumerable sets. Hence we consider nondeterministic multiple-reset machines that run in real time, in linear time, and in polynomial time. We show that linear time is no more powerful than real time and that a language can be accepted in linear time by a nondeterministic multiple-reset machine if and only if it can be accepted in real time by a nondeterministic machine with just three reset tapes as auxiliary storage. The class of languages accepted in real time by nondeterministic multiple-reset machines will be called MULTI-RESET.

Since the class of single-reset languages is similar in structure to the class of linear context-free languages, it is natural to compare the class MULTI-RESET with the class $\mathscr{L}_{BNP}$ of languages accepted in real time by nondeterministic reversal-bounded Turing machines. It is known that $\mathscr{L}_{BNP}$ is the smallest intersection-closed semi*AFL* containing *PAL* [6] and here we see that MULTI-RESET is the smallest intersection-closed semi*AFL* containing *COPY*. Although the class of single-reset languages is not comparable to the class of linear context-free languages, the language *COPY* is in $\mathscr{L}_{BNP}$ so that MULTI-RESET $\subseteq \mathscr{L}_{BNP}$. We conjecture that this inclusion is proper.

A language in MULTI-RESET is accepted in real time by a machine that makes a bounded number of "first-in, first-out'' comparisons, or resets. But we show in Section 3 that the class MULTI-RESET is closed under Kleene*—this means that (in a sense made precise in Sections 3 and 6) a nondeterministic machine that runs in linear time and makes an unbounded number of resets can be simulated by a nondeterministic machine that runs in real time and makes just one reset on each of its three tapes.

Our characterization of MULTI-RESET in terms of *COPY* leads to an algebraic characterization in terms of the regular sets: MULTI-RESET is the smallest class of languages containing the regular sets and closed under intersection and linear-erasing homomorphic duplication. From this fact we show that a wide range of classes of languages

specified by nondeterministic oracle machines can be characterized in terms of the regular sets and some variations of homomorphic duplication.

In Section 6 we consider languages representing "unbounded" comparisons: $COPY* = \{x_1x_1\$ \cdots x_mx_m\$ \mid m \geqslant 0,\ x_i \in \{a, b\}^*\}$ and $*COPY = \{(x\$)^m \mid m \geqslant 0,\ x \in \{a, b\}^*\}$. We provide characterizations of the semi*AFLs* generated by $COPY*$ and $*COPY$ in terms of machines with "reusable reset" tapes and "nonwriting circular" tapes. Further, we consider a language *SHIFT* that encodes certain sequences of pairs of strings, and characterize the semi*AFL* generated by *SHIFT* in terms of machines with "writing circular" tapes. Any language accepted in real time by a nondeterministic machine with reusable reset tapes or circular tapes is in fact a member of MULTI-RESET.

Three main themes are pursued in this paper: the study of automata, particularly restricted Turing machines; algebraic language theory and its interface with the study of computability and computational complexity; the modeling of different types of comparisons and their uses in terms of Turing machines. These themes interact with each other to produce the results

In Section 1 we review some basic concepts. The notion of single reset machines is developed in Section 2 and that of multiple-reset machines in Section 3. In Section 4, $\mathscr{L}_{BNP}$ and MULTI-RESET are compared, while homomorphic duplication and the relation between MULTI-RESET and other classes are studied in Section 5. Finally, reusable reset tapes and circular tapes are studied in Section 6.

1

It is assumed that the reader is familiar with the basic concepts from the theories of automata, computability, and formal languages. Some of the concepts that are most important for this paper are reviewed here and notation is established.

For a string w, $|\ w\ |$ denotes the *length* of w. We use e for the empty string. The *reversal* w^R of a string w is the string obtained by writing w in reverse order. For a string w, $w^1 = w$.

Recall that a *homomorphism* (between free monoids) is a function $h: \Sigma^* \to \Delta^*$ such that for all $x,\ y \in \Sigma^*$, $h(xy) = h(x)\,h(y)$. A homomorphism $h: \Sigma^* \to \Delta^*$ is *nonerasing* if $|\ w\ | > 0$ implies $|\ h(w)| > 0$ and is *length-preserving* if for all $w \in \Sigma^*$, $|\ h(w)| = |\ w\ |$. A homomorphism $h: \Sigma^* \to \Delta^*$ is *linear-erasing on language* $L \subseteq \Sigma^*$ if there is a constant $k > 0$ such that for all $w \in L$ with $|\ w\ | \geqslant k$, $|\ w\ | \leqslant k\ |\ h(w)|$, and is *polynomial-erasing on language* $L \subseteq \Sigma^*$ if there is a constant $k > 0$ such that for all $w \in L$ with $|\ w\ | \geqslant k$, $|\ w\ | \leqslant |\ h(w)|^k$. A class $\mathscr{L}$ of languages is *closed under* (*nonerasing*, *linear-erasing*, *polynomial-erasing*) *homomorphism* if for every language $L \in \mathscr{L}$ and any homomorphism h (that is nonerasing, linear-erasing on L, polynomial-erasing on L), $h(L) = \{h(w) \mid w \in L\}$ is in $\mathscr{L}$.

For a machine M, the language accepted by M is denoted by $L(M)$. A machine *operates in real time* if it reads a new input symbol at every step and it *operates in linear time* if there is a constant k such that any accepting computation on an input string of length n has at most kn steps. If f is a function, then a machine M *operates in time* $f(n)$ if any

accepting computation on an input string of length n has at most $f(n)$ steps. Let f be a function. Let $NTIME(f) = \{L(M) \mid M$ is a nondeterministic Turing machine that operates within time $f(n)\}$ and let $NP = \bigcup_{k\geq 1} NTIME(n^k)$, so that NP is the class of languages accepted in polynomial time by nondeterministic Turing machines.

A *semiAFL* is a family of languages containing at least one nonempty language and closed under the operations of union, inverse homomorphism, nonerasing homomorphism and intersection with regular sets. An *AFL* is a semi*AFL* that is closed under the operations of concatenation and Kleene*. A *full* semi*AFL* (*AFL*) is a semi*AFL* (*AFL*) closed under arbitrary homomorphism. A *principal* semi*AFL* (*AFL*) with generator L is the smallest semi*AFL* (*AFL*) containing L. We assume that the reader has some familiarity with these notions and with the connections between properties of classes of languages and properties of certain types of machines that specify these classes. See [8, 9, 13] for full developments.

For a class of languages $\mathscr{L}$, $\mathscr{M}(\mathscr{L})$ denotes the smallest semi*AFL* that contains $\mathscr{L}$ and $\mathscr{F}(\mathscr{L})$, the smallest *AFL* that contains $\mathscr{L}$. Similarly, $\mathscr{M}_\cap(\mathscr{L})$ ($\mathscr{F}_\cap(\mathscr{L})$) denotes the smallest intersection-closed semi*AFL* (respectively, *AFL*) that contains $\mathscr{L}$, and $\hat{\mathscr{M}}_\cap(\mathscr{L})$ ($\hat{\mathscr{F}}_\cap(\mathscr{L})$) denotes the smallest full intersection-closed semi*AFL* (*AFL*) that contains $\mathscr{L}$. When $\mathscr{L} = \{L\}$ consists of a single language, we write, e.g., $\mathscr{M}(L)$ for $\mathscr{M}(\mathscr{L})$.

The class $\mathscr{L}_{BNP}$ is defined as the class of languages accepted in linear time by nondeterministic Turing machines whose work tapes are reversal-bounded (i.e., there is a fixed constant that bounds the number of times a read-write head can change direction during any computation). It is known that a language L is in $\mathscr{L}_{BNP}$ if and only if there are three linear context-free languages L_1, L_2, L_3 and a nonerasing homomorphism h such that $L = h(L_1 \cap L_2 \cap L_3)$, and that $\mathscr{L}_{BNP}$ is the smallest intersection-closed semi*AFL* containing $\{ww^R \mid w \in \{a, b\}^*\}$ [6].

When x is a real number, $\lceil x \rceil$ denotes the smallest integer that is greater than or equal to x, and $\lfloor x \rfloor$, the largest integer that is less than or equal to x.

2

In this section we study "single-reset machines" and the class of languages specified by these machines. A single-reset machine is a Turing machine allowed only to make a certain "first-in, first-out" comparison.

A *reset tape* is a one-way infinite tape with one read-write head which moves only left-to-right and can be reset once to the left end of the tape.

Since only one reset is allowed on a reset tape and since the read-write head moves only left-to-right otherwise, we may assume that the tape squares read on the second sweep (i.e., after the reset is performed) are erased as the head moves over them. Now we consider the notion of an acceptor with a single reset tape as auxiliary work tape.

A nondeterministic *single-reset machine* is a nondeterministic acceptor with one input tape that is read in only one direction (say, from left to right), a finite-state control, and one reset tape as auxiliary work tape. The language accepted by a single-reset machine is a *single-reset language*.

We will consider only nondeterministic machines in this paper and thus we will not use the modifier "nondeterministic" except when emphasis is needed.

Based on our assumptions regarding the use of the reset tape, we can assume that an accepting configuration of a single-reset machine is reached only when the reset tape is empty and the machine is in an accepting state of its finite-state control.

The operation of a single-reset machine is analogous to that of a nondeterministic pushdown store acceptor whose read-write head on the pushdown store is allowed to make only one reversal. Recall that a language is linear context-free if and only if it is accepted by a pushdown store acceptor of this type and that this characterization leads to the following fact: a language L is linear context-free if and only if there exist homomorphisms h_1 and h_2 and a regular set S such that $L = \{h_1(w)\, h_2(w^R) \mid w \in S\}$. Let us compare the operation of a single-reset machine with the operation of a one-reversal pushdown store acceptor: both machines read input and write on the storage tape until at some point the read-write head on the reset tape resets to the left end of the tape and the read-write head on the pushdown store reverses direction; after the reset is made, the read-write head on the reset tape reads the string on the reset tape from left to right; after the reversal is made, the read-write head on the pushdown store reads the string on the pushdown store from right to left. This informal description leads to the following fact, the proof of which is omitted.

2.1. LEMMA. *A language L is a single-reset language if and only if there exist homomorphisms h_1 and h_2 and a regular set S such that $L = \{h_1(w)\, h_2(w) \mid w \in S\}$.*

We have placed no restrictions on the running time of a single-reset machine. Lemma 2.1 implies that the class of single-reset languages is closed under arbitrary homomorphic mappings. However, from the fact that the class of regular sets is closed under arbitrary homomorphic mappings, the characterization of Lemma 2.1 can be modified to require that the homomorphisms are at worst linear erasing on the regular set. This corresponds to requiring that single-reset machines operate in linear time. In fact, single-reset machines can be required to operate in real time.

2.2. LEMMA. *If M_1 is a single-reset machine, then there is a single-reset machine M_2 such that M_2 runs in real time and $L(M_2) = L(M_1)$.*

Proof. We provide only a sketch. A similar construction appears in [12].

If M_1 does not operate in real time, then in some computation either M_1 makes a series of writing moves on the reset tape while reading no new input symbols and this happens before the reset is made or M_1 makes a series of reading moves on the reset tape while reading no new input symbols and this happens after the reset is made, or both. Call a move in which no input is read an "e-move." To each string y that might be part of one written on the reset tape, we associate a table T_y that tells whether y can be written or read during e-moves and, if so, which states can begin and end such a sequence of moves. The set of such tables, for all possible y, must be finite (since the state set is finite) and for each table T the set $\{y \mid T = T_y\}$ is regular. The finite-state control of M_2 can therefore identify each possible table T and check if, for a given y, $T = T_y$. Instead of writing

a substring y on e-moves, M_2 attaches at the appropriate place an encoding of T_y; the finite-state control of M_2 can later decode this during the reading phase. Similarly, if M_2 guesses that a substring y will be read during e-moves only, it uses an encoding of T_y instead of y; if this guess is incorrect, M_2 can block during the reading phase. A similar idea applies to strings that are both read and written on e-moves. ▮

Notation. Let $COPY = \{ww \mid w \in \{a, b\}^*\}$ and let $PAL = \{ww^R \mid w \in \{a, b\}^*\}$.

It is easy to see that $COPY$ is a single-reset language. Since single-reset machines are nondeterministic and read input in one direction, it is easy to see that the class of single-reset languages is closed under union, inverse homomorphism, nonerasing homomorphism and intersection with regular sets. This leads to the following result.

2.3. THEOREM. *The class of single-reset languages is the smallest class containing COPY and closed under union, inverse homomorphism, nonerasing homomorphism, and intersection with regular sets, that is, the class of single-reset languages is a principal semiAFL with generator COPY. A language L is a single-reset language if and only if there exist homomorphisms h_1 and h_2 and a regular set R such that h_1 is nonerasing and $L = h_1(h_2^{-1}(COPY) \cap R)$.*

The proof of Theorem 2.3 is similar to the proof of the analogous characterization of the class of linear context-free languages as the principal semi*AFL* generated by *PAL* [9].

It is apparent that the class of single-reset languages is closed under reversal. From Lemma 2.1 one can show that this class is not closed under concatenation or Kleene*, and hence is not closed under substitution. From the definition of single-reset machines, one can obtain an intercalation lemma that yields the fact that $\{wcwcw \mid w \in \{a, b\}^*\}$ is not a single-reset language (or see [15]) and hence that the class of single-reset languages is not closed under intersection or complementation.

The preceding facts reveal a strong analogy between the class of single-reset languages and the class of linear context-free languages. However, these two classes are not comparable since $COPY$ is not a linear context-free language and PAL is not a single-reset language [15].

Now let us consider a generalization of the class of single-reset languages.

Let n be a positive integer and let ρ be a function from $\{1,..., n\}$ to $\{1, R\}$. Let L be a language and let $h_1,..., h_n$ be homomorphisms. The language $\langle \rho; h_1,..., h_n \rangle (L) = \{h_1(w)^{\rho(1)} \cdots h_n(w)^{\rho(n)} \mid w \in L\}$ is a *homomorphic replication of type ρ on L*; it is a *homomorphic duplication of L* if $\rho(i) = 1$ *for each i.*

Consider the smallest class of languages containing the regular sets and closed under homomorphic replication. This class of languages is precisely the class of languages accepted by nondeterministic one-way checking-stack automata whose checking-stack head is reversal-bounded [12]. We shall refer to this class as $\mathscr{L}_{REP}$. If instead of homomorphic replication we consider homomorphic duplication, then we have a different situation. The smallest class of languages containing the regular sets and closed under homomorphi-duplication is precisely the class of languages accepted by nondeterministic one-way checking-stack automata whose checking-stack head reads the checking stack in only one

direction and can perform a finite number (independent of the input) of resets [12]. We shall refer to this class as $\mathscr{L}_{DUP}$.

Since homomorphic duplication is a restriction of homomorphic replication, it is clear that for any class $\mathscr{C}$ of languages the closure of $\mathscr{C}$ under homomorphic duplication is contained in the closure of $\mathscr{C}$ under homomorphic replication. In particular, $\mathscr{L}_{DUP} \subseteq \mathscr{L}_{REP}$. Results in [15] show that this inclusion is proper. The class $\mathscr{L}_{DUP}$ is also the class of equal-matrix languages of [16].

Notice that $COPY = \langle \rho; h_1, h_2 \rangle (\{a, b\}^*)$ where $\rho(1) = \rho(2) = 1$ and h_1, h_2 are both the identity homomorphism. From Lemma 2.1, we see that a language L is single-reset if and only if there is a regular set R and homomorphisms h_1 and h_2 such that $L = \langle \rho; h_1, h_2 \rangle (R)$, where $\rho(1) = \rho(2) = 1$. Thus $\mathscr{L}_{DUP}$ can also be characterized as the closure of the class of single-reset languages under homomorphic duplication. This leads to the following fact.

2.4. PROPOSITION. *If $\mathscr{L}$ is a class of languages that contains all of the regular sets and is closed under homomorphic duplication, then every single-reset language is in $\mathscr{L}$.*

3

Now we consider multitape machines where each auxiliary work tape is a reset tape. The class of languages accepted by such machines (in real time) has a simple characterization in terms of *COPY*; other properties of this class of languages are established. In Sections 4 and 5 we will examine the relationship of this class to a variety of other classes of languages that have been studied in the literature.

A *multiple-reset* machine is an acceptor with a one-way input tape, finite-state control, and some finite number of reset tapes as auxiliary work tapes. The first result shows that nondeterministic multiple-reset machines are sufficiently powerful to perform any (effective) computation.

3.1. THEOREM. *A language is recursively enumerable if and only if it is accepted by a nondeterministic acceptor with a one-way input tape and two reset tapes as auxiliary storage.*

Proof. Let L_1 be a recursively enumerable language and let M_1 be a deterministic one-head, one-tape Turing machine such that $L(M_1) = L_1$. From M_1 construct an acceptor M_2 with a one-way input tape and two reset tapes such that M_2 operates in the following way. On reading input string w, M_2 writes the initial instantaneous description ID_0 of M_1's computation on w on one of its tapes, say Tape 1. Then M_2 begins to guess a sequence of instantaneous descriptions of M_1, writing the sequence on both Tape 1 and Tape 2 separated by markers. At some point M_2 may guess an instantaneous description that indicates that M_1 is in an accepting mode. At this point M_2 has a string of the form $ID_0 \# ID_1 \# \cdots \# ID_t \#$ on Tape 1 (for some $t > 0$) and the string $ID_1 \# \cdots \# ID_t \#$ on Tape 2, where ID_0 is the initial instantaneous description of M_1 on the input and ID_t is accepting. M_2 now resets the read-write heads on both Tapes 1 and 2 and reads

the tapes in synchrony, checking as it does so that for $1 \leqslant i \leqslant t$, ID_i (on Tape 2) results from ID_{i-1} (on Tape 1) under the rules of M_1. If each check succeeds, then $ID_0, \ldots, ID_t$ represents an accepting computation of M_1 on the input w and so M_2 accepts w. If at any time a guess is judged to be incorrect, then M_2 halts in a nonaccepting state. Thus M_2 is a nondeterministic multiple-reset machine and $L(M_2) = L(M_1) = L_1$. Note that if M_1 operates in time $f(n)$ then for each string w in L_1 there is an accepting computation of M_2 on input w with at most $2(f(|w|) + 1)^2$ steps. ∎

From Theorem 3.1 we see that a class of multiple-reset machines must be restricted in some way if the languages accepted by the machines in the class are required to be recursive. Therefore we consider time-bounded multiple-reset machines, restricting attention in this section to the cases of real time and linear time.

Consider a nondeterministic multiple-reset machine that operates in real time. If it has k reset tapes as work tapes, then the "multitape representation theorem" of [13] (see Lemma 1.2 of [6] for a simple version and informal proof) shows how to characterize the language accepted by that machine in terms of k single-reset languages.

3.2. THEOREM. *For every $k \geqslant 1$, a language L is accepted in real time by a nondeterministic acceptor with k reset tapes as work tapes if and only if there exist k single-reset languages $L_1, \ldots, L_k$ and a length-preserving homomorphism h such that L is the image of the intersection of $L_1, \ldots, L_k$ under h, i.e., $L = h(L_1 \cap \cdots \cap L_k)$.*

From Theorem 3.1 and the "multitape representation theorem", we have the following fact.

3.3. THEOREM. *A language L is recursively enumerable if and only if there exist two single-reset languages L_1 and L_2 and a homomorphism h such that $h(L_1 \cap L_2) = L$.*

Let MULTI-RESET denote the class of languages accepted in real time by nondeterministic multiple-reset machines.

From the representation of languages accepted in real time by multiple-reset machines given in Theorem 3.2, various properties of the class MULTI-RESET follow from general results [8, 13]. Some of these properties are summarized in the following way.

3.4. THEOREM. *The class* MULTI-RESET *is closed under union, intersection, inverse homomorphism, nonerasing homomorphism, and reversal. It is the smallest semiAFL that is closed under intersection and contains the language COPY.*

For certain types of multitape acceptors linear time is no more powerful than real time (e.g., nondeterministic Turing machines [4] and nondeterministic reversal-bounded machines [6]). It is natural to ask whether the class of nondeterministic multiple-reset machines has the same property. The answer is affirmative.

3.5. THEOREM. *If M_1 is a multiple-reset machine that operates in linear time, then there is a multiple-reset machine M_2 such that M_2 operates in real time and $L(M_2) = L(M_1)$. Thus, for the class of multiple-reset machines, linear time is no more powerful than real time.*

Proof. Let M_1 be a multiple-reset machine that has k single-reset tapes as work tapes and that runs in time tn where $t > 1$. From M_1 we construct a machine M_2 with $k + 4$ single-reset tapes as work tapes. Each computation of M_2 will attempt to simulate an accepting computation of M_1 and will use four tapes, say Tapes 1–4, for "bookkeeping" purposes and will use k tapes, say Tapes 5-$k + 4$, to simulate the k tapes of M_1. During an accepting computation, described below, M_2 guesses a compressed version of its input string, tests whether M_1 accepts it and verifies that the guessed and actual input are the same.

Let $p = 2t + 4$.

Phase 1. Initially, M_2 reads its input tape and copies this input onto Tape 1. However, M_2 stores p symbols per tape square on Tape 1 while reading one symbol per tape square from the input tape. Simultaneously, M_2 nondeterministically guesses a string, say $b_1 \cdots b_r$, and stores it on each of Tapes 2, 3, and 4, writing p symbols on each tape square. In addition, M_2 guesses a distinguished position in the copies of this string on Tapes 3 and 4 and marks that symbol. Let us say that the marked symbol is b_j. (The marked symbol is a guess of the middle of the computation.) After making the guess of $b_1 \cdots b_r$, the read-write heads on Tapes 2–4 are reset. During this phase there is no activity on Tapes 5-$k + 4$; the total time for this phase is $\lceil r/p \rceil$ steps.

Phase 2. Using the contents of Tape 2 (the compressed version of $b_1 \cdots b_r$) as input, M_2 now uses Tapes 5-$k + 4$ to simulate a computation of M_1. Since one tape square of Tape 2 contains p symbols, this simulation can be "sped up" so that the total time is at most $\lceil tr/p \rceil$. During this phase M_2 continues to read input at the rate of one symbol per step and to store this input on Tape 1 with p symbols being stored in each tape square. If the computation of M_1 that is simulated is an accepting computation (so that M_1 accepts $b_1 \cdots b_r$), then M_2 goes on to Phase 3; otherwise, M_2 transfers into a nonaccepting "dead" state.

Phase 3. Recall that the contents of Tapes 3 and 4 are identical, both containing $b_1 \cdots b_r$ with p symbols per tape square and the symbol b_j marked. To begin this phase, M_2 reads Tape 3 one tape square per step until the square containing the marked symbol is scanned. Until that square is scanned, M_2 continues to read from its input tape and store that input on Tape 1, still compressed by p. When M_2 scans the tape square of Tape 3 containing b_j, then M_2 ceases to copy the input onto Tape 1 and begins to compare the remaining input with the remainder $b_{j+1} \cdots b_r$ of the "guessed" string on Tape 3. At that point the read-write head of Tape 1 is reset and the contents of Tapes 1 and 4 are compared one tape square per step. If the contents of Tapes 1 and 4 agree up to the marked symbol b_j, then the actual input string has $b_1 \cdots b_j$ as a prefix. If the contents of Tape 3 agrees with the remainder of the input, then the entire input string is $b_1 \cdots b_r$. Since $b_1 \cdots b_r$ is accepted by M_1, M_2 accepts.

Clearly M_2 is a multiple-reset machine. Since M_2 was forced to read a new input symbol at each step, M_2 operates in real time. We must verify that M_2 has enough time in an accepting computation to make the guess, simulate M_1 on the guess, and check whether the guess is correct.

Assuming that $b_1 \cdots b_r$ is the actual input to M_2, Phase 1 takes $\lceil r/p \rceil$ steps and Phase 2 takes at most $\lceil tr/p \rceil$ steps. During Phase 3 M_2 must move across Tape 3 until the square containing b_j is scanned and this action takes $\lceil j/p \rceil$ steps. Then M_2 must compare Tapes 1 and 4 before the remainder of the input is read. Comparing Tapes 1 and 4 takes $\lceil j/p \rceil$ steps. Thus the following inequality must be satisfied:

$$\lceil r/p \rceil + \lceil tr/p \rceil + 2\lceil j/p \rceil \leqslant r.$$

It is also necessary that the jth symbol of the actual input not be read until the head on Tape 3 reaches the marked symbol in Phase 3, or

$$\lceil r/p \rceil + \lceil tr/p \rceil + \lceil j/p \rceil \leqslant j.$$

For $j = \lfloor r/2 \rfloor$ and $p = 2t + 4$, these inequalities are satisfied for all but a finite number of input strings.

The machine M_2 is a nondeterministic multiple-reset machine that operates in real time such that $L(M_2) = L(M_1)$. ∎

From Theorem 3.5 the following fact is immediate.

3.6. COROLLARY. *The class* MULTI-RESET *is closed under linear erasing.*

The fact that MULTI-RESET is closed under linear erasing, or equivalently, that linear time is no more powerful than real time for the class of multiple-reset machines, suggests that MULTI-RESET may have another interesting property. Recall that, for example, a language L is accepted in linear time by a nondeterministic multitape Turing machine if and only if L is accepted in real time by a nondeterministic Turing machine with just three pushdown stores as work tapes [4]. If the original linear-time machine is reversal-bounded, then the real time machine is also reversal-bounded [6]. The analogous statement about multiple reset machines is also true.

3.7. THEOREM. *Let L be a language. The following are equivalent:*

(i) *L is accepted in linear time by a nondeterministic multiple-reset machine;*

(ii) *L is accepted in real time by a nondeterministic machine with three single-reset tapes as storage tapes;*

(iii) *There exist single-reset languages $L_1, \ldots, L_k$ and a homomorphism h such that h is linear-erasing on $L_1 \cap \cdots \cap L_k$ and $h(L_1 \cap \cdots \cap L_k) = L$;*

(iv) *There exist single-reset languages L_1, L_2, L_3 and a nonerasing homomorphism h such that $h(L_1 \cap L_2 \cap L_3) = L$.*

The proof of Theorem 3.7 is deferred until Section 6; Theorem 3.7 is a corollary of Theorem 6.5.

Notation. Let $COPY * = \{x_1x_1\$ \cdots x_mx_m\$ \mid m \geqslant 0, \text{ each } x_i \in \{a, b\}^*\}$.

3.8. THEOREM. *The class* MULTI-RESET *is closed under Kleene* and under e-free substitution.*

Proof. If MULTI-RESET is closed under Kleene*, then with the properties it already possesses by virtue of Theorem 3.4 MULTI-RESET is an *AFL*. An *AFL* that is closed under intersection is also closed under *e*-free substitution, so it suffices to show that MULTI-RESET is closed under Kleene*.

Since MULTI-RESET is the smallest semi*AFL* that is closed under intersection and contains the language *COPY*, to show that MULTI-RESET is closed under Kleene* it is sufficient to show that the language $COPY*$ is in MULTI-RESET [9].

Consider a nondeterministic acceptor M that operates in the following way. On reading input $w \in \{a, b, \$\}^*$, M reads a portion $u_1 \in \{a, b\}^*$ of the input and copies this onto Tape 1. Guessing that u_1 is the "first half" of a string in *COPY*, M reads the next portion $v_1 \in \{a, b\}^*$ of the input and copies v_1 onto Tape 2. When M encounters the first occurrence of \$, it copies \$ onto both Tapes 1 and 2 and treats the next portion $u_2 \in \{a, b\}^*$ of the input as if it were the "first half" of a string in *COPY* and copies u_2 onto Tape 1. Then M reads the next portion $v_2 \in \{a, b\}^*$ of the input, recording v_2 on Tape 2 until the next occurrence of \$ is read. Continuing in this way, M has written $u_1\$\, u_2\$ \cdots u_m\$$ on Tape 1 and $v_1\$\, v_2\$ \cdots v_m\$$ on Tape 2 when the entire input string has been read and the input string was $u_1v_1\$ \cdots u_mv_m\$$. At this time M resets the heads of Tapes 1 and 2 and reads the two tapes simultaneously, attempting to match each string u_i on Tape 1 with the corresponding v_i on Tape 2. If for each $i = 1,\ldots, m$, M finds that $u_i = v_i$, then M accepts the input string.

Clearly M is a multiple-reset machine that runs in time $2n$ and $L(M) = COPY*$. Since MULTI-RESET is closed under linear erasing, this means that $COPY*$ is in MULTI-RESET. ∎

From Theorems 3.7 and 3.8 we see that a real time nondeterministic multiple-reset machine can simulate another that makes an unbounded number of independent "comparisons" by making only one reset on each of three tapes.

The reader should note that all of the properties of MULTI-RESET described in this section with the exception of closure under Kleene* and *e*-free substitution are also properties of the class $\mathscr{L}_{BNP}$. A direct comparison of MULTI-RESET and $\mathscr{L}_{BNP}$ is made in the next section.

4

In this section we point out some relationships between the class of single-reset languages and certain other classes of languages.

Recall from Section 2 that the class of single-reset languages is the smallest (full) semi*AFL* containing the language *COPY*. Hence we refer to this class as $\mathscr{M}(COPY)$. Similarly the class of linear context-free languages is the smallest (full) semi*AFL* containing the language *PAL*, and so we refer to this class as $\mathscr{M}(PAL)$.

It is known [15] that the classes $\mathscr{M}(COPY)$ and $\mathscr{M}(PAL)$ are not comparable since $PAL \notin \mathscr{M}(COPY)$ and $COPY \notin \mathscr{M}(PAL)$. Thus we are interested in the intersection of

these two classes. Clearly, $\{a^nb^n \mid n \geqslant 0\}$ is in $\mathscr{M}(PAL) \cap \mathscr{M}(COPY)$ and so $\mathscr{M}(\{a^nb^n \mid n \geqslant 0\}) \subseteq \mathscr{M}(PAL) \cap \mathscr{M}(COPY)$.

The class $\mathscr{M}(\{a^nb^n \mid n \geqslant 0\})$ is the class of languages accepted by nondeterministic machines with a one-way input tape and auxiliary storage consisting of a single counter that makes at most one reversal or, equivalently in the case of counters, makes at most one reset. A simple "information-content" argument shows that $COPY \notin \mathscr{M}(\{a^nb^n \mid n \geqslant 0\})$ and so we have the following fact.

4.1. PROPOSITION. $\mathscr{M}(\{a^nb^n \mid n \geqslant 0\}) \subsetneq \mathscr{M}(COPY)$.

We do not know whether the inclusion $\mathscr{M}(\{a^nb^n \mid n \geqslant 0\}) \subseteq \mathscr{M}(COPY) \cap \mathscr{M}(PAL)$ is proper or whether there are any context-free languages in $\mathscr{M}(COPY) - \mathscr{M}(\{a^nb^n \mid n \geqslant 0\})$. We conjecture that both answers are "no."

The smallest intersection-closed semi*AFL* containing $\{a^nb^n \mid n \geqslant 0\}$ is closed under arbitrary homomorphic mappings [17] but not under Kleene* [1]. Thus, $\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}) \subsetneq \mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}))$. We now show that this implies that $\mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}))$ is not closed under intersection, that is, $\mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\})) \neq \mathscr{F}_\cap(\mathscr{M}(\{a^nb^n \mid n \geqslant 0\}))$.

A class $\mathscr{C}$ of languages is *translatable* if for every $L \in \mathscr{C}$ and every choice of two symbols c, d that do not occur in any string in L, the language $\{c^nwd^n \mid n \geqslant 0,\ w \in L\}$ is in $\mathscr{C}$.

Clearly any intersection-closed semi*AFL* that contains $\{a^nb^n \mid n \geqslant 0\}$ is translatable, so both $\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\})$ and $\mathscr{F}_\cap(\mathscr{M}(\{a^nb^n \mid n \geqslant 0\}))$ are translatable. It is known [2, 10] that if $\mathscr{C}$ is a semi*AFL* with the property that $\mathscr{F}(\mathscr{C})$ is translatable, then $\mathscr{C}$ is an *AFL*, that is, $\mathscr{C} = \mathscr{F}(\mathscr{C})$. Since $\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}) \neq \mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}))$, we see that $\mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}))$ is not translatable. Hence, $\mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\})) \neq \mathscr{F}_\cap(\mathscr{M}(\{a^nb^n \mid n \geqslant 0\}))$ since the latter class is translatable and the former is not.

From the definitions of the classes, we see that $\mathscr{F}_\cap(\{a^nb^n \mid n \geqslant 0\}) = \mathscr{F}_\cap(\mathscr{M}(\{a^nb^n \mid n \geqslant 0\})$. Since $\{a^nb^n \mid n \geqslant 0\}$ is a single-reset language and MULTI-RESET is an *AFL* closed under intersection, we see that $\mathscr{F}_\cap(\{a^nb^n \mid n \geqslant 0\}) \subseteq$ MULTI-RESET. Again, using a simple "information-content" argument, one can show that *COPY* is not in $\mathscr{F}_\cap(\{a^nb^n \mid n \geqslant 0\})$ so that $\mathscr{F}_\cap\{(a^nb^n \mid n \geqslant 0\}) \neq$ MULTI-RESET.

4.2. PROPOSITION. *$\mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}))$ is not translatable and $\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}) \subsetneq \mathscr{F}(\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\})) \subsetneq \mathscr{F}_\cap(\{a^nb^n \mid n \geqslant 0\}) \subsetneq$* MULTI-RESET.

How do MULTI-RESET and $\mathscr{L}_{BNP}$ compare? Since $\mathscr{L}_{BNP}$ is the smallest intersection-closed semi*AFL* containing the language *PAL*, we see that $\mathscr{L}_{BNP} \subset$ MULTI-RESET if and only if $PAL \in$ MULTI-RESET and that MULTI-RESET $\subseteq \mathscr{L}_{BNP}$ if and only if $COPY \in \mathscr{L}_{BNP}$. It is easy to construct a nondeterministic reversal-bounded machine that runs in linear time and recognizes *COPY* so that MULTI-RESET $\subseteq \mathscr{L}_{BNP}$. As shown in Section 3, MULTI-RESET is closed under Kleene* but it is conjectured in [5] that $\mathscr{L}_{BNP}$ is not closed under Kleene*. We conjecture that $PAL \notin$ MULTI-RESET and that MULTI-RESET $\subsetneq \mathscr{L}_{BNP}$.

Summing up the inclusions discussed above we have the following fact.

4.3. PROPOSITION. MULTI-RESET $\subseteq \mathscr{L}_{BNP}$

Recall that a language is context-free if and only if it is accepted in real time by a nondeterministic pushdown store acceptor. Let D be a nondeterministic pushdown store acceptor that runs in real time and let w be a string accepted by D. For any accepting computation of D on w, there is a sequence $y_1, y_2, \ldots, y_{|w|}$ of strings representing the successive pushdown store contents in this computation, and for each y_i in this sequence, $|y_i| \leqslant |w|$. Hence a nondeterministic machine M with two single-reset tapes as auxiliary storage can simulate computations of D by first "guessing" the sequence of pushdown store contents and then "checking" to see if each guess of a string in the sequence follows from the previous string in a computation of D. Then $L(M) = L(D)$ and an accepting computation of M on a string w takes at most $2|w|^2$ steps. Combining this argument with "compression" techniques yields the following fact.

4.4. PROPOSITION. *If L is a context-free language, then there is a nondeterministic machine M with two single-reset tapes as auxiliary storage such that $L(M) = L$ and M accepts in time n^2. Equivalently, for every context-free language L_1 there is a language $L_2 \in$* MULTI-RESET *and a homomorphism h such that $h(L_2) = L_1$ and for all $w \in L_2$, $|w| \leqslant |h(w)|^2$.*

Recall that a *quasi-realtime* language is one accepted in real time by a nondeterministic multitape Turing machine, and that a language L is quasi-realtime if and only if there exist context-free languages L_1, L_2, L_3 and a nonerasing homomorphism h such that $h(L_1 \cap L_2 \cap L_3) = L$ [4]. Let $\mathscr{L}$ be the class of languages accepted by nondeterministic machines with some finite number of single-reset tapes as auxiliary storage and which operate in time n^2. Clearly $\mathscr{L}$ is closed under intersection and nonerasing homomorphism, so that from Proposition 4.4 we conclude that every quasi-realtime language is in $\mathscr{L}$.

The image of the class of quasi-realtime languages under polynomial-erasing homomorphism is clearly equal to the class NP of languages accepted in polynomial time by nondeterministic Turing machines. Thus the image of the class MULTI-RESET under polynomial-erasing homomorphism is the class NP. Similarly the image of the class MULTI-RESET under arbitrary homomorphic mappings is the class RE of recursively enumerable (r.e.) sets. Note that $\mathscr{L}_{BNP}$ has these same properties, i.e., the image of $\mathscr{L}_{BNP}$ under polynomial-erasing homomorphism is NP and the image of $\mathscr{L}_{BNP}$ under arbitrary homomorphism is the class of r.e. sets.

Notation. For any class $\mathscr{C}$ of languages, $H_{\text{poly}}(\mathscr{C})$ is the image of $\mathscr{C}$ under polynomial-erasing homomorphism and $\hat{H}(\mathscr{C})$ is the image of $\mathscr{C}$ under arbitrary homomorphic mappings.

4.5. PROPOSITION. H_{poly}(MULTI-RESET) $= H_{\text{poly}}(\mathscr{L}_{BNP}) = NP$ *and* $\hat{H}$(MULTI-RESET) $= \hat{H}(\mathscr{L}_{BNP}) = RE$.

It is known that the smallest intersection-closed full semi*AFL* containing $\{a^n b^n \mid n \geqslant 0\}$ is not closed under Kleene*. Thus in contrast with Propositions 4.1–4.3 we have the following fact.

4.6. PROPOSITION. $\mathscr{M}_\cap(\{a^nb^n \mid n \geqslant 0\}) \subsetneq \mathscr{F}_\cap(\{a^nb^n \mid n \geqslant 0\}) = \mathscr{M}_\cap(COPY) = \mathscr{M}_\cap(PAL) = RE$.

5

In this section MULTI-RESET is characterized algebraically in terms of a restricted form of homomorphic duplication. This leads to considering classes of languages specified by certain types of Turing machines.

A class $\mathscr{C}$ of languages is *closed under nonerasing* (*linear-erasing*, *polynomial-erasing*) *homomorphic duplication* if for every $L \in \mathscr{C}$, every $n \geqslant 1$, and every n homomorphisms $h_1, \ldots, h_n$ each of which is nonerasing (respectively, linear-erasing on L, polynomial-erasing on L), the language $\langle \rho; h_1, \ldots, h_n \rangle (L)$ is in $\mathscr{C}$ where $\rho(i) = 1$ for each i.

From Proposition 2.4 and Theorem 3.4, we obtain a characterization of the class MULTI-RESET that is similar to the characterization of $\mathscr{L}_{BNP}$ given by Theorem 4.1 of [3].

5.1. THEOREM. *The class* MULTI-RESET *is the smallest class of languages containing all of the regular sets and closed under intersection and linear-erasing homomorphic duplication. It is the smallest nonempty class that is closed under intersection, inverse homomorphism, intersection with regular sets, and nonerasing homomorphic duplication, that is, the class* MULTI-RESET *is the smallest semiAFL containing* $\{e\}$ *that is closed under intersection and nonerasing homomorphic duplication.*

Proof. Clearly, MULTI-RESET contains all of the regular sets and is closed under intersection. Based on the proof of Theorem 4.1 of [3], it is easy to see that MULTI-RESET is closed under linear-erasing homomorphic duplication.

Let $\mathscr{L}$ be the smallest class of languages containing all of the regular sets and closed under intersection and linear-erasing homomorphic duplication. From the remarks above, $\mathscr{L} \subseteq$ MULTI-RESET. From Proposition 2.4 we see that every single-reset language is in $\mathscr{L}$. From Theorem 3.7 we see that every language in MULTI-RESET can be expressed as the nonerasing homomorphic image of three single-reset languages. Since $\mathscr{L}$ is closed under intersection and nonerasing homomorphism, every language in MULTI-RESET is in $\mathscr{L}$.

The other characterizations are obtained similarly, using Theorem 2.3. ∎

As was discussed in Section 4, the smallest intersection-closed full semi*AFL* containing the language *COPY* is the class of r.e. sets, and the closure of MULTI-RESET under arbitrary homomorphic mappings is the class of r.e. sets. Using this fact and Theorem 5.1, we obtain the following result.

5.2. COROLLARY. *The class of r.e. sets is the smallest class of languages containing the regular sets and closed under intersection and homomorphic duplication. It is the smallest nonempty class that is closed under intersection, inverse homomorphism, intersection with regular sets, and homomorphic duplication, that is, the class of r.e. sets is the smallest (full) semiAFL that is closed under intersection and homomorphic duplication.*

Similarly, recall from Section 4 that the class NP is the smallest semiAFL containing $COPY$ and closed under intersection and polynomial-erasing. This leads to the following result.

5.3. COROLLARY. *The class NP is the smallest class of languages containing the regular sets and closed under intersection and polynomial-erasing homomorphic duplication. It is the smallest nonempty class that is closed under intersection, inverse homomorphism, intersection with regular sets, and polynomial-erasing homomorphic duplication, that is, the class NP is the smallest semiAFL that is closed under intersection and polynomial-erasing homomorphic duplication.*

Theorem 5.1 and Corollaries 5.2 and 5.3 are similar to the results developed in [3] for $\mathscr{L}_{BNP}$ and the operation of homomorphic replication.

Now we consider "oracle" machines.

An *oracle machine* is a multitape Turing machine M with a distinguished work tape, the *query* tape, and three distinguished states, $q_?$, q_{yes}, q_{no}. At any step of a computation on an input string w, M may transfer into the state $q_?$. From state $q_?$ M transfers into the state q_{yes} if the string currently appearing on the query tape is in a given *oracle set* A; otherwise, M transfers into the state q_{no}; in either case the query tape is (instantly) erased. The set of strings *accepted by M relative to the oracle set A* is $L(M, A) = \{w \mid$ there is an accepting computation of M on input w when the oracle set is $A\}$.

The following "representation lemma" is established in [7].

5.4. LEMMA. *Let M be an oracle machine that runs in time $t(n)$ and has tape alphabet Δ. There exist homomorphic h_1 and h_2 and a language L_M such that (i) for any oracle set $A \subseteq \Delta^*$, $L(M, A) = h_1(L_M \cap h_2^{-1}((cA \cup d(\Delta^* - A))^*))$ where c and d are two symbols not in Δ,* (ii) *L_M is accepted in linear time by a deterministic multitape Turing machine, and* (iii) *for all $w \in L_M$, $| w | \leqslant t(| h_1(w) |)$.*

This lemma and Theorem 5.1 provide the basis for characterizations of several classes of languages specified by oracle machines. First, consider the class of languages that are "recursively enumerable in A."

For any language A, the class $RE(A)$ of languages that are *recursively enumerable in A* is the class of all languages accepted by unrestricted oracle machines that have oracle set A.

Notation. If A is a language and Σ is the smallest (finite) alphabet such that $A \subseteq \Sigma^*$, then let $\bar{A} = \Sigma^* - A$. If $A, B \subseteq \Sigma^*$, then let $A \oplus B = cA \cup dB$ where c, d are two symbols not in Σ.

5.5. THEOREM. *For any language A, the class $RE(A)$ is the smallest class of languages containing the regular sets and the language $(A \oplus \bar{A})^*$ and closed under intersection, inverse homomorphism, and homomorphic duplication. It is the smallest class containing $(A \oplus \bar{A})^*$ that is closed under intersection, inverse homomorphism, intersection with regular sets, and homomorphic duplication, that is, the class $RE(A)$ is the smallest (full) semiAFL containing $(A \oplus \bar{A})^*$ that is closed under intersection and homomorphic duplication.*

Proof. For any language A, let $\mathscr{L}(A)$ be the smallest class of languages containing the regular sets and $(A \oplus \bar{A})^*$ and closed under intersection, inverse homomorphism, and homomorphic duplication.

It is clear that $RE(A)$ contains $(A \oplus \bar{A})^*$ and the regular sets and that $RE(A)$ is closed under intersection, inverse homomorphism, and homomorphic duplication. Since $\mathscr{L}(A)$ was chosen to be the smallest such class, $\mathscr{L}(A) \subseteq RE(A)$.

Consider any $L \in RE(A)$. From Lemma 5.4 we see that there exist homomorphisms h_1 and h_2 and a language L_M such that $L = h_1(L_M \cap h_2^{-1}((A \oplus \bar{A})^*))$ and L_M is accepted in linear time by a deterministic multitape Turing machine. Since $\mathscr{L}(A)$ contains the regular sets and is closed under intersection and homomorphic duplication, it follows from Corollary 5.3 that every r.e. set is in $\mathscr{L}(A)$; hence, $L_M \in \mathscr{L}(A)$. By definition, $(A \oplus \bar{A})^*$ is in $\mathscr{L}(A)$ and $\mathscr{L}(A)$ is closed under inverse homomorphism so that $h_2^{-1}((A \oplus \bar{A})^*) \in \mathscr{L}(A)$. Since $\mathscr{L}(A)$ is closed under intersection, $L_M \cap h_2^{-1}((A \oplus \bar{A})^*) \in \mathscr{L}(A)$. Since $\mathscr{L}(A)$ is closed under homomorphic duplication, $h_1(L_M \cap h_2^{-1}((A \oplus \bar{A})^*)) \in \mathscr{L}(A)$ so that $L \in \mathscr{L}(A)$. Since L was taken arbitrarily from $RE(A)$, this shows that $RE(A) \subseteq \mathscr{L}(A)$.

The other characterizations follow immediately. ∎

5.6. THEOREM. *For any language A the class $NP(A)$ of languages that are nondeterministic polynomial time in A is the smallest class of languages containing $(A \oplus \bar{A})^*$ and the regular sets and closed under intersection and polynomial-erasing homomorphic duplication.*

Theorems 5.5 and 5.6 are similar to results in [3, 7]. Characterizations similar to those in [3, 7], but using homomorphic duplication instead of homomorphic replication, can be developed for the class of arithmetical sets, the class of rudimentary languages the class of extended rudimentary languages, the class of elementary languages, the class of primitive recursive languages, the class of recursive languages, and their appropriate relativizations.

We conclude this section with several results describing the closure under intersection and homomorphic duplication of certain classes of languages.

5.7. THEOREM. *Let $\mathscr{C}$ be a class of languages containing the regular sets and closed under inverse homomorphism and marked concatenation. The class $\{h(L_1 \cap L_2) \mid L_1 \in \mathscr{C}$, $L_2 \in$ MULTI-RESET, and h is a nonerasing homomorphism$\}$ is the smallest class containing all languages in $\mathscr{C}$ and all regular sets and closed under intersection and linear-erasing homomorphic duplication.*

Sketch of the proof. The techniques needed are those used in [5]. The following facts lead to the proof. Let $\mathscr{L} = \{h(L_1 \cap L_2) \mid L_1 \in \mathscr{C}$, $L_2 \in$ MULTI-RESET, and h is a nonerasing homomorphism$\}$.

(1) If $L \in \mathscr{C}$ and h is a homomorphism that is linear-erasing on L, then there exist homomorphisms f and g with f length-preserving, and a language $L_2 \in$ MULTI-RESET such that $h(L) = f(g^{-1}(L) \cap L_2)$.

(2) For any $k \geqslant 1$ and any choice of $L_1, \ldots, L_k \in \mathscr{C}$, there exist a language $C \in \mathscr{C}$, single-reset languages M_1 and M_2, and a homomorphism h such that $h(C \cap M_1 \cap M_2) = L_1 \cap \cdots \cap L_k$ and h is linear-erasing on $C \cap M_1 \cap M_2$.

(3) The class $\mathscr{L}$ is closed under intersection and linear-erasing homomorphism.

(4) The class $\mathscr{L}$ is closed under linear-erasing homomorphic duplication. ∎

5.8. COROLLARY. *Let $\mathscr{C}$ be a class of languages containing the regular sets and closed under inverse homomorphism and marked concatenation. The class $\{h(L_1 \cap L_2) \mid L_1 \in \mathscr{C}$, $L_2 \in$ MULTI-RESET, and h is a homomorphism (that is polynomial-erasing on $L_1 \cap L_2)\}$ is the smallest class containing all languages in $\mathscr{C}$ and all regular sets and closed under intersection and homomorphic duplication (resp., polynomial-erasing homomorphic duplication).*

5.9. COROLLARY. *Any semiAFL containing COPY and closed under intersection is closed under linear erasing.*

6

Recall from Theorem 3.8 that MULTI-RESET is closed under Kleene*, even though any multiple-reset machine has only a fixed number of tapes and each tape can make only one reset. Since accepting a language L^* given a machine for L might involve checking segments of the input independently and there is no bound on the number of segments, there should be a generalization of reset tapes (and corresponding generalization of *COPY*, the generator of MULTI-RESET) that removes the restriction to one reset per tape, allowing an unbounded number of "comparisons," but which has no greater power when multiple tapes are allowed.

The obvious method, in this context, for generalizing *COPY* is to take a Kleene* or marked Kleene* and this is the language $COPY* = \{x_1x_1\$ \cdots x_mx_m\$ \mid m \geqslant 0, x_i \in \{a, b\}^*\}$. If we interpret *COPY* as signifying "compare one pair of strings," then $COPY*$ signifies "compare many pairs of strings, independently." The corresponding storage structure, as can be deduced from general principles [9], is one that acts like a reset tape between "initializations."

A *reusable reset tape* is a reset tape for which, in addition, the head may be repositioned to the left end of the tape. This action causes the entire tape to be erased, reinitializing it to the empty tape.

To extend the definitions of single-reset and multiple-reset machines, one considers nondeterministic machines with a one-way input tape, a finite state control and reusable reset tapes as auxiliary storage. No bound is placed on the number of times the head on a reusable tape can be repositioned; an accepting configuration is reached when the work tapes are empty and the machine is in an accepting state.

The following characterizations are straightforward applications of basic results in *AFL* theory.

6.1. PROPOSITION. (1) *The class of languages accepted in real time by nondeterministic*

machines with one reusable reset tape is precisely the semiAFL generated by COPY∗ (i.e., the smallest class of languages containing COPY∗ and closed under nonerasing homomorphism, inverse homomorphism, union, and intersection with regular sets), and the AFL generated by COPY.

(2) *The class of languages accepted in real time by nondeterministic machines with multiple reusable reset tapes is precisely the intersection-closed semiAFL generated by COPY∗.*

Now *COPY* can also be interpreted as signifying "test that two strings are equal." It can therefore be generalized by a language that tests that "many strings are equal."

Notation. Let $*COPY = \{(x\$)^m \mid m \geqslant 0, x \in \{a, b\}^*\}$.

The storage structure corresponding to $*COPY$ is a tape that can only be read and that is read from left to right in full sweeps.

A *circular tape* has a read-write head that moves only from left to right but may be reset to the left end when the right end is reached; the string written on the tape during the first pass determines the length of tape used thereafter. If the instructions allow symbols on the tape to be changed then the tape is *writing* and otherwise it is *nonwriting*. (Thus a nonwriting circular tape is like a checking stack with restricted head movement.) Since the head is required to make full sweeps, an accepting configuration for a circular tape is reached only when the head is on the rightmost symbol of the tape contents.

Note that a reset tape is essentially a nonwriting circular tape on which the head makes only two sweeps.

6.2. PROPOSITION. (1) *The class of languages accepted in real time by nondeterministic machines with one nonwriting circular tape as work tape is precisely the semiAFL generated by $*COPY$.*

(2) *The class of languages accepted in real time by nondeterministic machines with multiple nonwriting circular tapes as work tapes is precisely the intersection-closed semiAFL generated by $*COPY$.*

The analog of Proposition 6.2 for writing circular tapes uses a language in which the strings to be compared are overlapped in a certain way.

Notation. (1) For strings $x, y \in S^*$ of the same length, the parallel pairing of x with y is a string $\langle x, y\rangle$ in $(S \times S)^*$: if $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_n$ with $n \geqslant 0$, each x_i and y_i in S, then $\langle x, y\rangle = [x_1, y_1] \cdots [x_n, y_n]$ where $[x_i, y_i] \in S \times S$.

(2) Let $SHIFT = \{\langle w_1, w_2\rangle \$ \langle w_2, w_3\rangle \$ \cdots \$ \langle w_{m-1}, w_m\rangle \$ \mid m \geqslant 2, w_i \in \{a, b\}^*, \mid w_1 \mid = \mid w_2 \mid = \cdots = \mid w_m \mid\} \cup \{e\}$.

6.3. PROPOSITION. (1) *The class of languages accepted in real time by nondeterministic machines with one writing circular tape as work tape is precisely the semiAFL generated by SHIFT.*

(2) *The class of languages accepted in real time by nondeterministic machines with multiple writing circular tapes as work tapes is precisely the intersection-closed semiAFL generated by SHIFT.*

As was seen in Section 3, *COPY** is in MULTI-RESET; it is not difficult to see that $*COPY$ and *SHIFT* can be accepted by nondeterministic machines, each with two reset tapes, that operate in linear time, so also $*COPY$ and *SHIFT* are in MULTI-RESET. (One reset tape is not sufficient, as is discussed in [14].) Hence the classes of languages accepted in linear time by nondeterministic machines with multiple reusable reset tapes or circular tapes are in fact all equal to MULTI-RESET. Based on results in Section 5 we see that any language in *NP* can be accepted in polynomial time by a nondeterministic machine with some number of nonwriting circular tapes, and any recursively enumerable set can be accepted by a nondeterministic machine with some number of nonwriting circular tapes (that operates without time bound). Analogous statements hold for reusable reset tapes and writing circular tapes.

Use of the storage structures described above allows us to prove the result on which Theorem 3.7 (three single-reset tapes are sufficient for a multiple-reset machine) is based. The result follows once the next lemma is established.

6.4. LEMMA. *Suppose* $k \geqslant 1$ *and* $L_1, \ldots, L_k$ *are single-reset languages. Then there is a nondeterministic machine with one reset tape and one nonwriting circular tape that accepts* $L_1 \cap \cdots \cap L_k$ *in real time.*

Proof. For $1 \leqslant i \leqslant k$ let M_i be a single-reset machine accepting $L_i \subseteq \Sigma^*$ in real time. We may assume that M_i writes at most one symbol on its tape in any step.

We will describe the operation of a nondeterministic machine M with a nonwriting circular tape (called Tape 1) and a reset tape (Tape 2) such that M operates in real time and $L(M) = L_1 \cap \cdots \cap L_k$. On its first sweep of Tape 1, M will write a string with $2k + 4$ tracks, of the following form. The first track contains a string xy where x (respectively, y) is a string from Σ^* compressed to $4k + 4$ symbols per square, with the last symbol in x (respectively, y) possibly containing $6k + 6$ symbols from Σ (respectively, between $2k + 2$ and $8k + 7$ symbols). The string xy will be used as input to $M_1, \ldots, M_k$ so k of the symbols from Σ within xy are marked to indicate the steps in which the single-reset machines make their resets. The second track on Tape 1 contains, *right justified*, a string $\bar{x}$ of the same form as x with $|y| - 1 \leqslant |\bar{x}| \leqslant |y|$. The remaining tracks contain $w_1, \ldots, w_{2k+2} \in \Sigma^*$ with $|w_i| = |xy|$. Thus Tape 1 contains

$$\begin{array}{cc} x & y \\ & \bar{x} \\ w_1 & \\ \vdots & \\ w_{2k+2} & \end{array}$$

where x, y, $\bar{x}$ are compressed.

Let u be the homomorphism that "uncompresses" strings, so that, e.g., $u(xy) \in \Sigma^*$.

The operation of M has five phases.

Phase 1. To begin, M makes k sweeps of Tape 1. In the ith sweep, M checks that its actual input is w_i and uses xy as the input to M_i, simulating M_i on $u(xy)$ up to the

step in which it would reset and writing a string z_i on Tape 2 to record the string M_i would write on its tape. Each z_i is compressed (to correspond to the compression of xy) and has its last symbol to separate it from z_{i+1}. After the kth sweep Tape 1 is reset.

Phase 2. In this phase M checks that the next portion of the input is w_{k+1} while copying the first two tracks of Tape 1 onto Tape 2. Tape 2 will then contain

$$z_1 z_2 \cdots z_k \begin{matrix} x & y. \\ \bar{x} & \end{matrix}$$

Both tapes are now reset.

Phase 3. Now M makes another k sweeps of Tape 1. In the ith sweep, M compares w_{i+k+1} to the actual input being read and uses the end of xy (after the marker for M_i's reset) and the string z_i written in Phase 1 to simulate the behavior of M_i after its reset, checking that an accepting state is reached. Tape 1 is reset after the kth sweep.

Phase 4. Tape 2 now contains only

$$\begin{matrix} x & y \\ \bar{x}. & \end{matrix}$$

M checks that the input being read is w_{2k+2} and, when the start of $\bar{x}$ on Tape 1 is reached, compares that $\bar{x}$ to the x written on Tape 2, to check that $x = \bar{x}$; Tape 1 is then reset.

Phase 5. Finally, M makes $2k + 2$ sweeps of Tape 1 and finishes reading Tape 2 and the input, checking that $w_1 \cdots w_{2k+2} = u(\bar{x})$ and that the remaining input is $u(y)$ (using $\bar{x}$ and y on Tape 2). If all the comparisons and the computations of the M_i's have been successful, then M accepts the input.

Note that M makes only $4k + 4$ sweeps of Tape 1.

To see that M accepts exactly $L_1 \cap \cdots \cap L_k$, first suppose that M accepts an input v after writing xy, $\bar{x}$, and $w_1, \ldots, w_{2k+2}$ on Tape 1. Then $v = w_1 \cdots w_{2k+2} u(y)$, and $u(\bar{x}) = w_1 \cdots w_{2k+2}$ (from Phase 5) and $x = \bar{x}$ (from Phase 4), so $v = u(xy)$. But from Phases 1 and 3, each M_i accepts $u(xy)$, so $v \in L_1 \cap \cdots \cap L_k$. Conversely, if $v \in L_1 \cap \cdots \cap L_k$ then it is possible to construct properly compressed strings x, y such that $u(xy) = v$, $|y| - 1 \leqslant |x| \leqslant |y|$, $|xy| = \lfloor |v|/4k + 4 \rfloor$, and $u(x) = w_1 \cdots w_{2k+2}$ with $|w_i| = \lfloor |v|/4k + 4 \rfloor = |xy|$. Then M can accept v by writing xy, $\bar{x} = x$, and $w_1, \ldots, w_{2k+2}$ on Tape 1. ∎

6.5. THEOREM. *For any language L the following are equivalent.*

(1) $L \in$ MULTI-RESET.

(2) *L can be accepted in real time by a nondeterministic machine with one reset tape and one nonwriting circular tape.*

(3) *L is the image under a nonerasing homomorphism of the intersection of a single-reset language and a language in the semiAFL generated by $*COPY$.*

(4) *L can be accepted in real time by a nondeterministic machine with three reset tapes.*

(5) *L is the image under a nonerasing homomorphism of the intersection of three single-reset languages.*

Proof. Standard techniques can be used to show that (2) is equivalent to (3) and that (4) is equivalent to (5) [8, 13]. Clearly (5) implies (1). If $L \in$ MULTI-RESET, then L is the image under a nonerasing homomorphism of the intersection of some number of single-reset languages, so from Lemma 6.4 and the fact that the class of languages accepted by the hybrid machines of (2) is closed under nonerasing homomorphism, (1) implies (2). To see that (3) implies (4), note that $*COPY$ can be accepted in real time by a nondeterministic machine with *two* reset tapes, and so any language in $\mathscr{M}(*COPY)$ can be so accepted. A simple construction then shows that the class of languages described in (4) contains any language that is the intersection of a single-reset language and a language in $\mathscr{M}(*COPY)$. Since the class of languages in (4) is closed under nonerasing homomorphism, it follows that (3) implies (4). ∎

REFERENCES

1. B. BAKER AND R. BOOK, Reversal-bounded multipushdown machines, *J. Comput. System Sci.* **8** (1974), 315–332.
2. L. BOASSON AND M. NIVAT, Sur diverses familles de langages fermées par transduction rationnelle, *Acta Informatica* **2** (1973), 180–188.
3. R. BOOK, Simple representations of certain classes of languages, *J. Assoc. Comput. Mach.* **25** (1978), 23–31.
4. R. BOOK AND S. GREIBACH, Quasi-realtime languages, *Math. Systems Theory* **4** (1970), 97–111.
5. R. BOOK AND M. NIVAT, Linear languages and the intersection closures of classes of languages, *SIAM J. Comput.* **7** (1978), 167–177.
6. R. BOOK, M. NIVAT, AND M. PATERSON, Reversal-bounded acceptors and intersections of linear languages, *SIAM J. Comput.* **3** (1974), 283–295.
7. R. BOOK AND C. WRATHALL, On languages specified by relative acceptance, *Theor. Comput. Sci.* **7** (1978), 185–195.
8. S. GINSBURG AND S. GREIBACH, Abstract families of languages, "Studies in Abstract Families of Languages," Memoir No. 87, Amer. Math. Soc., Providence, R. I., 1969.
9. S. GINSBURG AND S. GREIBACH, Principal AFL, *J. Comput. System Sci.* **4** (1970), 308–338.
10. S. GREIBACH, Erasing in context-free AFLs, *Inform. Contr.* **21** (1972), 436–465.
11. S. GREIBACH, Control sets on context-free grammar forms, *J. Comput. System Sci.* **15** (1977), 35–98.
12. S. GREIBACH, One-way finite visit automata, *Theor. Comput. Sci.* **6** (1978), 175–222.
13. S. GREIBACH AND S. GINSBURG, Multi-tape AFA, *J. Assoc. Comput. Mach.* **19** (1972), 193–221.
14. S. GREIBACH AND C. WRATHALL, in preparation.
15. K. KLINGENSTEIN, ρ-matrix languages, *Theor. Comput. Sci.*, in press.
16. R. SIROMONEY, Finite-turn checking automata, *J. Comput. System Sci.* **5** (1971), 549–559.
17. S. GREIBACH, Remarks on blind and partially blind one-way multicounter machines, *Theor. Comput. Sci.* **7** (1978), 311–324.