Closure Properties of Certain Classes of Languages under Generalized Morphic Replication

Z. FANG[1] AND J. S. DEOGUN[2]

[1] *Computer Science Department, The Wichita State University, Wichita, KS 67208, U.S.A.*

[2] *Computer Science Department, University of Nebraska, Lincoln, Nebraska 68588, U.S.A.*

In this paper a new language operator, a generalized morphic replication, is introduced. Let Ω be a finite set of morphisms and reversal morphisms from Σ^* into Δ^*, and ω be in Ω^*. A morphic replicator is defined as follows: for each x in Σ^* define $\omega(x)$ to be $h_1(x) \ldots h_m(x)$, where $|\omega| = m$ and $\omega = h_1 \ldots h_m$. A generalized morphic replication extends ω to languages by $\omega(L) = \{\omega(x): x \text{ is in } L\}$ and to sets W of morphic replicators, where $W \subseteq \Omega^*$, $W(x) = \{\omega(x): \omega \text{ is in } W\}$ and $W(L) = \bigcup W(x)$, where the union is taken over all x in L.

It is shown that the class of languages accepted in real time by a non-deterministic reversal-bounded multitape Turing machine, the class of NP, and the class of the recursively enumerable sets, are all closed under the generalized morphic replication when the morphisms and the reversal morphisms are, respectively, linear-erasing, polynomial-erasing, and arbitrary.

Received September 1985, revised February 1987

1. INTRODUCTION

In the study of formal languages and abstract automata one can notice that almost all of the language operations, such as the Boolean operations, morphism, inverse morphism, concatenation, Kleene* and reversal preserve the class of regular sets.[7] However, the operation of morphic replication defined in Book[3] does not preserve the class of regular sets. In fact, the morphic replication is the concatenation of images of n morphisms or reversal morphisms of the same string w, $w \in L$. The concept of morphic replication can be generalized as an operation in which the string of morphisms or reversal morphisms is replaced by a language, possibly an infinite set. We call this operation a generalized morphism replication.

In this paper we define the concept of generalized morphism replication and study the closure properties of certain classes of languages under this operation. Specifically, it is shown that the class of languages accepted in real time by the non-deterministic reversal-bounded multi-tape Turing machine $\mathsf{L}_{\mathrm{BNP}}$,[2,4] the class of NP, and the class of the recursively enumerable sets are closed under generalized morphism replication when the morphism and the reversal morphisms are, respectively, linear-erasing, polynomial-erasing, or arbitrary.

2. PRELIMINARIES

In this section, preliminary concepts relevant to generalized morphic replication are reviewed and the notations used in this paper are established. Then the concept of generalized morphic replication is defined and some primitive features of this operation are discussed.

For a string w, $|w|$ denotes the length of w. The reversal reversal w^R of a string w is the string obtained by writing w in reverse order.

A morphism (between two free monoids) is a function $h: \Sigma^* \to \Delta^*$ such that for all $x, y \in \Sigma^*$, $h(xy) = h(x)\,h(y)$. A reversal morphism for a given morphism is a function $g^R: \Sigma^* \to \Delta^*$ such that for all $x \in \Sigma^*$, $g^R(x) = [g(x)]^R$. Furthermore, a morphism, $h: \Sigma^* \to \Delta^*$, is nonerasing if $|w| > 0$ implies $|h(w)| > 0$ and is length-preserving if $\forall\ w \in \Sigma^*$, $|h(w)| = |w|$. Given a language $L \subseteq \Sigma^*$, the morphism h is linear-erasing on L 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 L 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 L of languages is closed under any morphism (non-erasing, linear-erasing, polynomial-erasing) if, for every $L \in \mathsf{L}$ and any morphism h (that is non-erasing, linear-erasing, or polynomial-erasing on L), $h(L) = \{h(w) \mid w \in L\} \in \mathsf{L}$.

A generalized morphic replication is defined as follows.

Definition. Let Ω be a finite set of morphisms and reversal morphisms from Σ^* into Δ^*. We call ω a *morphic replicator* where ω is in Ω^*. Then for each x in Σ^* define $\omega(x)$ to be $h_1(x) \ldots h_m(x)$, where $|\omega| = m$ and $\omega = h_1 \ldots h_m$. If $m = 0$ or $x = \lambda$, were λ is the empty symbol, then $\omega(x) = \lambda$ also.

Generalized morphic replication is defined to extend ω to languages L by $\omega(L) = \{\omega(x): x \text{ is in } L\}$ and to languages W of morphic replicators, where $W \subseteq \Omega^*$, $W(x) = \{\omega(x): \omega \text{ is in } W\}$ and $W(L) = \bigcup W(x)$, where the union is taken over all x in L.

The concept of morphic replication by Book[3] is based on a single fixed string of n morphisms or reversal morphisms. In this paper we extend the concept of replication by replacing a fixed single string with a language, which could be an infinite set.

The $W(L)$ defined above is called the generalized morphism replication of a non-erasing (or linear-erasing, or polynomial-erasing) morphism, if every element in Ω is a non-erasing (or linear-erasing, or polynomial-erasing) morphism of L.

Let both L and W be two languages in a class of languages L, and let Ω be the alphabet set of W, such that each element in Ω is a non-erasing (or linear-erasing, or polynomial-erasing) morphism or reversal morphism on L. Then if $W(L) \in \mathsf{L}$, L is said to be closed under the generalized morphic replication of a non-erasing (or linear-erasing, or polynomial-erasing) morphism.

Let $L = \Sigma^*$ be a regular set. If $\Omega = \{g, h\}$, where $g(p) = 0$ and $h(p) = 1$ for all $p \in \Sigma$, then $W = (ghg)^*$

is a regular set.[7] However, $W(L) = \{(ghg)^*(w) \mid w \in \Sigma^*\} = \{0^n 1^n 0^n 0^n 1^n 0^n \ldots 0^n 1^n 0^n \mid n \geqslant 0\}$ is a context-sensitive language.[7] Therefore, it may be noted that the class of regular sets is not closed under generalized morphic replication of a length-preserving morphism.

3. SEVERAL IMPORTANT LEMMAS

In this section we present some preliminary lemmas, proofs of Theorem 3.1 of this section and the main theorems of Section 4 are based on these lemmas.

Lemma 3.1. A language L is linear context-free if and only if there is a regular set R and two linear-erasing morphisms h_1 and h_2, such that $L = \{h_1(w)[h_2(w)]^R \mid w \in R\}$.[5,6]

Lemma 3.2. For every recursively enumerable set L, there exist two linear context-free languages L_1 and L_2, and a morphism h, such that $L = \{h(w) \mid w \in L_1 \cap L_2\}$.[1]

Lemma 3.3. For every $k \geqslant 1$, a language L is accepted by a non-deterministic Turing acceptor M with k pushdown stores as auxiliary storage which operates in real time if and only if there exist k deterministic context-free languages $L_1, \ldots, L_k$ and a length-preserving morphism h such that L is the image of the intersection of $L_1, \ldots, L_k$ under h, $L = h(L_1 \cap \ldots \cap L_k)$. Further, if the ith pushdown store of $M (1 \leqslant i \leqslant k)$ is restricted in its operation (e.g. is reversal-bounded or is a counter), then the language L_i can be accepted by a deterministic pushdown store acceptor N_i which operates in real time and whose pushdown store is restricted in the same way as the ith pushdown store of M.[2]

Lemma 3.4. Let L be a language. The following are equivalent:

(i) L is accepted by a non-deterministic multi-pushdown acceptor which operates in such a way that, in every computation, each pushdown store makes, at most, a bounded number of reversals and runs in linear time.

(ii) L is accepted by a non-deterministic multi-pushdown acceptor which operates in such a way that, in every computation, each pushdown store makes, at most, one reversal and runs in real time.

(iii) L is the length-preserving morphic image of the intersection of some finite number of linear context-free languages.

(iv) L is accepted by a non-deterministic acceptor with three pushdown stores which operate in such a way that, in every computation, each pushdown store makes, at most, one reversal and runs in real time.

(v) L is the length-preserving morphic image of the intersection of three linear context-free languages.[2]

It is well known that the computation of a single Turing machine tape can be imitated by two pushdown stores without any loss of time. If the Turing machine tape makes r reversals, then each pushdown store will also make, at most, r reversals.[2] Similarly, a multi-tape Turing machine can be simulated by a multi-pushdown acceptor without any loss of time or an increase in the number of reversals. Thus, in part (i) and (ii) of Lemma 3.4 the phrase 'multi-pushdown acceptor' can be replaced by 'multi-tape acceptor', and the statement that (i)–(v) are equivalent is still true. In the remainder of this paper only the term multi-tape acceptor will be used. The class characterized in Lemma 3.4 will be referred to as $\mathsf{L}_{\mathrm{BNP}}$.

Theorem 3.1. Let N be the set of natural numbers and Σ be a finite alphabet. There exist two linear context-free languages L_1 and L_2 and a length-preserving morphism h such that $\{(wc)^k \mid w \in \Sigma^*, k \in N\} = h(L_1 \cap L_2)$, were c is a symbol not in Σ.

Proof. To prove the theorem we construct a non-deterministic on-line acceptor M such that $L(M) = \{(wc)^k \mid w \in \Sigma^*, k \in N\}$, M operates in real time and M has two pushdown stores, each of which makes only one reversal. Therefore, for this acceptor, Lemma 3.3 guarantees the existence of L_1, L_2 and h.

The acceptor M has two pushdown stores referred to as tape 1 and tape 2. Each tape has two tracks. During the computation M writes a sequence of 'blocks' on each tape. A block consists of the two strings u and v stored on the top and bottom tracks of tapes 1 and 2. Strings u and v are of the same length, $|u| = |v|$, and are terminated by endmarkers after the last symbol in each string. The end of strings u and v marks the end of the block. M reads its input at the rate of one symbol per step of its computation in order to operate in real time. Let $(wc)^k$ be the input string of symbols that can be accepted by M, where $w \in \Sigma^*$, $k \in N$.

We define two numbers $n = \lceil k - 2/2 \rceil$ and $m = \lfloor |w|/2 \rfloor$, not known to the acceptor M, to prove the theorem.

An accepting computation of M can be described as in the following three phases.

Phase 1. M reads its input at the rate of one symbol per step and writes the input symbol on the top track of tapes 1 and 2. Each time M reads a symbol, M non-deterministically guesses and writes an arbitrary symbol from Σ on the bottom track of each tape. M writes the same symbol on the bottom track of tapes 1 and 2. If the input symbol is the endmarker symbol 'c', then M writes a 'c' on both the top and bottom tracks of each tape. Each block consists of a pair of strings, $u_i c$ and $v_i c$, in $\Sigma^* c$ such that $|u_i| = |v_i|$ and $u_i c$ is the string tape. Each block consists of a pair of strings, $u_i c$ and $v_i c$, in $\Sigma^* c$ such that $|u_i| = |v_i|$ and $u_i c$ is the string read from the input and $v_i \in \Sigma^*$.

After reading a block, M either non-deterministically repeats the actions in phase 1 or moves to phase 2. However, during the computation in which M accepts its input, after having written such n blocks, M moves to phase 2. In the case that M moves to phase 2 after it has written more or less than n blocks, the computation of M is non-accepting. That is seen in phase 3.

Phase 2. In phase 2, M reads one input symbol per step, writes it on the top track of tapes 1 and 2, and non-deterministically writes an arbitrary symbol from Σ on the bottom track of each tape just as in phase 1. After M reads the portion x of input in this phase, M non-deterministically begins to perform the following actions.

Let y be the string on the bottom track of each tape when M has read portion x of the input. Clearly, $|x| = |y|$. Then M reads a symbol $a \in \Sigma$ and copies it onto the top and bottom tracks of tape 1, while the head of tape 2 doesn't act if $|w|$ is odd. If $|w|$ is even, M goes to next step immediately. In the next step M starts to pop tape 2, copying the symbols from x and y onto the bottom and top tracks, respectively, of tape 1. Simultaneously, M continues to read the input symbols and matches the symbols popped from the bottom track of tape 2 with the actual input read. If any comparison doesn't agree, M halts in a non-accepting state. If the next input symbol is 'c' and the symbol popped from tape 2 is also 'c', M

copies it onto both tracks of tape 1. At that time $|x| = m$, and M moves to phase 3. If M reads 'c', but the symbol popped from tape 2 is not 'c', M halts in a non-accepting state. After this operation, the contents of the tapes are as follows:

If $|w|$ is odd, tape 1: $u_1 cu_2 c \ldots cu_n cxay^R c$
$v_1 cv_2 c \ldots cv_n cyax^R c$
tape 2: $u_1 cu_2 c \ldots cu_n c$
$v_1 cv_2 c \ldots cv_n c$

If $|w|$ is even, tape 1: $u_1 cu_2 c \ldots cu_n cxy^R c$
$v_1 cv_2 c \ldots cv_n cyx^R c$
tape 2: $u_1 cu_2 c \ldots cu_n c$
$v_1 cv_2 c \ldots cv_n c$

where the rightmost 'c' is being scanned. If all the matched symbols agree, then the input read so far is recorded on the top track of tape 1.

Phase 3. In phase 3, M pops tapes 1 and 2 simultaneously, comparing respective blocks symbol by symbol. Success of all these comparisons implies:

$$u_1 = u_2 = \ldots = u_n = xy^R = (yx^R)^R = (v_n)^R = \ldots = (v_2)^R = (v_1)^R$$

if $|w|$ is even and

$$u_1 = u_2 = \ldots = u_n = xay^R = (yax^R)^R = (v_n)^R = \ldots = (v_2)^R = (v_1)^R$$

if $|w|$ is odd.

At the same time, M continues to read the remaining input and compares it with the bottom track of tape 1. M halts in a non-accepting state if any comparison doesn't succeed, if the input is exhausted before tape 1 or tape 2 is empty, or if tape 1 and tape 2 are empty before the input is exhausted (i.e. in phase 1 M would have moved to phase 2 after having read more or less than n blocks.) If these comparisons all agree and the input is exhausted, then tape 1 is empty if k is odd, or tape 2 is empty if k is even, and M halts in an accepting state. In that case, the input to M was $(wc)^k$, $w \in \Sigma^*$ and $w = xy^R$ if $|w|$ is even or $w = xay^R$ if $|w|$ is odd. Clearly, M satisfies the requirements of Lemma 3.3.■

4. MAIN RESULTS

In this section we will prove the closure property of certain classes of languages under generalized morphic replication using the lemmas established in Section 3.

Theorem 4.1. The $\mathsf{L}_{\mathrm{BNP}}$ is closed under the generalized morphic replication of a linear erasing morphism.

Proof. Let L and W be two languages in $\mathsf{L}_{\mathrm{BNP}}$ and let Σ be the alphabet of L, and Ω the alphabet of W. Assume Ω is a finite set of linear-erasing morphisms and reversal morphisms on L. Then, $L \subseteq \Sigma^*$, for every $\omega \in \Omega$, $\omega: \Sigma^* \to \Delta^*$. And there is a constant $k > 0$ such that for all $\omega \in \Omega$, $x \in L$, with $|x| \geqslant k$, then $|x| \leqslant k|\omega(x)|$.

$W(L) = \{\omega(x) \mid \omega \in W, x \in L\}$, if $\omega = h_1 \ldots h_n$, $x = p_1 \ldots p_m$, $\omega(x) = (h_1 \ldots h_n)(p_1 \ldots p_m) = h_1(p_1) \ldots h_1(p_m) \ldots h_n(P_1) \ldots h_n(p_m)$. We want to prove that $W(L)$ is in $\mathsf{L}_{\mathrm{BNP}}$ also.

Since L and W both are in $\mathsf{L}_{\mathrm{BNP}}$, there are two nondeterministic acceptors, M_L and M_W, each with three pushdown stores as auxiliary storage. M_L and M_W operate in real time, each pushdown store makes, at most, one reversal, and $L(M_L) = L$, $L(M_W) = W$. From Lemma 3.4 and its description, if we can construct a nondeterministic multi-tape Turing machine TM such that TM operates in linear time and each tape makes at most a bounded number of reversals and $W(L) = L(TM)$, then $W(L)$ is in $\mathsf{L}_{\mathrm{BNP}}$ and the theorem is true.

The principal problem is how TM can recognize $\omega(x)$, i.e. $h_1(p_1 \ldots p_m) h_2(p_1 \ldots p_m) \ldots h_n(p_1 \ldots p_m)$ while using only bounded reversals per tape. Therefore, the problem is to find the string of morphisms and reversal morphisms which acts on x, where $x \in L$. We suggest that $x^{|\omega|}$ be stored on one tape and $h_1^{|x|} h_2^{|x|} \ldots h_n^{|x|}$ be stored on another tape. These two tapes then act on each other and TM matches the result with the input string. It is rather easy to recognize $h_1^{|x|} h_2^{|x|} \ldots h_n^{|x|}$. In order to recognize $x^{|\omega|}$ we must use the technique which was used in proving Theorem 3.1.

Now we construct such a Turing machine. The Turing machine TM has an input tape that is read from left to right, a finite-state control unit, and ten storage tapes referred to as tape 1, tape 2, ..., tape 10.

Tape 1 is used to store the actual input y in order to be matched with the guessed input $\omega(x)$ and to control TM's running time which must be less than $t|y|$ where t is a constant. Tape 2 is used to guess $x^{|\omega|}$ nondeterministically. It is divided into two tracks. On the top track $x^{|\omega|}$ is stored and the bottom track has only one symbol 'c' which is placed on the cell corresponding to the last symbol of x in the top track. The other cells on the bottom track are empty. Tape 3 is used to store $(x^{|\omega|})^R$. Its role is the same as tape 2 when h is a reversal morphism. Tape 4 stores $h_1^{|x|} h_2^{|x|} \ldots h_n^{|x|}$. Like tape 2, tape 4 is also divided into two tracks. On the top track of tape 4 the string $h_1^{|x|} \ldots h_n^{|x|}$ is stored. On the bottom track a 'c' is placed on each cell corresponding to the $|x|$th h, and all other cells are empty. Tape 5 is used to store the guessed input $\omega(x)$ which is compared with the string in tape 1. Tapes 6 and 7 are used to simulate two pushdown stores of the acceptor M as in Theorem 3.1. The endmarker 'c' is, however, written on the bottom track of tape 2 at the cell corresponding to the last symbol of each instance of the string x instead of following x as in Theorem 3.1. Tapes 6 and 7 are respectively divided into three tracks each. Tracks one and two play the same role as the top and bottom tracks of the two tapes used in Theorem 3.1, and the third track is used to write the symbol 'c'. Tapes 8, 9 and 10 are used to simulate three pushdown stores M_L and M_W. The purpose of tapes 8, 9 and 10 is to decide the inclusion of x in L and that of ω in W.

Since every morphism in Ω is linear-erasing, there is a constant $k > 0$ for all $h \in \Omega$ and $x \in L$ with $|x| \geqslant k$, such that $|x| \leqslant k|h(x)|$. Now if the input string y is equal to $\omega(x) = h_1(x) h_2(x) \ldots h_n(x)$, then $|y| = |h_1(x) h_2(x) \ldots h_n(x)| = |h_1(x)| + |h_2(x)| + \ldots |h_n(x)|$, so that $k|y| = k|h_1(x)| + k|h_2(x)| + \ldots k|h_n(x)| \geqslant n|x|$. The length of string that we store in tape 2 is equal to $|x^{|\omega|}| = |x^n| = n|x|$ and the length of tape 4 is equal to $|h_1^{|x|} h_2^{|x|} \ldots h_n^{|x|}| = n|x|$. It may be noted that the length of tapes 2, 3 and 4 is less than or equal to $k|y|$.

The accepting computations of TM are described below.

Phase 1. During phase 1, TM reads input string $y \in \Delta^*$ and copies it into tape 1. After the input tape is exhausted the read–write head of tape 1 returns to its original position (so that it is in the leftmost cell of tape

1). In this phase the read–write head of tape 1 makes one reversal and the time required is $2|y|$.

Phase 2. During phase 2, *TM* non-deterministically guesses and writes one arbitrary symbol per step from Σ on the top track of tape 2. *TM* non-deterministically guesses a symbol from {blank, endmarker 'c'} and simultaneously writes it on the bottom tracks of tapes 2 and tape 4. *TM* non-deterministically guesses a symbol from Ω and repeatedly writes this symbol on the top track of tape 4 until an endmarker symbol appears on the bottom track. (This symbol is also written in the cell on the top track of tape 4 corresponding to the cell containing the endmarker symbol on the bottom track of tape 4). After having written 'c' on the bottom track of both tapes 2 and 4, *TM* non-deterministically repeats the actions in Phase 2 or moves to Phase 3. In its finite control, *TM* keeps a count k of steps and once every k steps, the read–write head of tape 1 moves a cell to the right. If the head of tape 1 moves into a blank cell, *TM* halts in a non-accepting state. In this phase no reversals take place and the time required is at most $k|y|$. After finishing this phase, the contents of tape 2 and tape 4 are as follows:

Tape 2: $p_{11}p_{12}\cdots p_{1m_1}p_{21}p_{22}\cdots p_{2m_2}\cdots p_{q1}\cdots p_{qm_q}$
$b\ \ b\ \ \ldots c\ \ \ b\ \ b\ \ \ldots c\ \ \ \ldots b\ \ \ldots c$

Tape 4: $h_1h_1\ldots h_1h_2h_2\ldots h_2\ldots h_q\ldots h_q$
$b\ \ b\ \ \ldots c\ b\ \ b\ \ \ldots c\ \ \ldots b\ \ \ldots c$

where the heads of tapes 2 and 4 are scanning the rightmost symbol.

Phase 3. The heads of tapes 1, 2 and 4 are returned to the original leftmost position. Thus tapes 1, 2 and 4 make one reversal just before the computation starts in phase 3. The *TM* simulates a computation of the acceptor in Theorem 3.1 of the contents of tape 2 as input and uses tapes 6 and 7 to simulate two pushdown stores to check whether or not the contents of tape 2 is x^q, where $x\in\Sigma^*$, and $q\in N$. If the computation is accepted, *TM* turns into phase 4, otherwise *TM* halts in a non-accepting state. During this phase the heads of tapes 1, 2, 4, 6 and 7 make exactly one reversal and the time required is $2k|y|$.

Phase 4. As *TM* moves into phase 4, the head of tape 2 is in the rightmost position and the head of tape 3 is in the leftmost position. As the computation begins, the head of tape 2 moves from right to left and that of tape 3 moves from left to right. The contents on the top track of tape 2 are read from right to left and are written on tape 3 from left to right. When the head of tape 2 has returned to its leftmost position, *TM* writes $(x^q)^R$ on tape 3. Then the head of tape 3 is returned. During this phase the head of tape 3 makes one reversal and the time required is $2k|y|$.

Phase 5. In phase 5, *TM* reads the contents on the top track of tape 2 until the first 'c' appears on the bottom track, and *TM* simulates a computation of M_L using tapes 8, 9 and 10 to simulate three reversal-bounded pushdown stores to check whether or not x is in L. If the computation is accepted, then the read–write head of tape 3 is returned to its original position and *TM* turns into phase 6. Otherwise *TM* halts in a non-accepting state. In this phase *TM* moves $2k|y|$ steps at most and the heads of tapes 2, 8, 9 and 10 make exactly one reversal.

Phase 6. In phase 6, the symbols in the cells on the top track of tape 4 corresponding to the cells in which the endmarker appears on the bottom track of tape 4 are read as input. Therefore, input to phase 6 is $\omega = h_1\ldots h_q$. *TM* simulates a computation of M_W using tapes 8, 9 and 10 to analog M_W's three pushdown stores to check whether or not $\omega = h_1\ldots h_q$ is in W. If the computation is accepted, then the read–write head of tape 4 is returned to its original position and *TM* moves to phase 7. Otherwise, *TM* halts in a non-accepting state. In this phase *TM* moves $2k|y|$ steps at most and the heads of tapes 4, 8, 9 and 10 make exactly one reversal.

Phase 7. During phase 7, *TM* reads symbols simultaneously from tapes 1, 2, 3, 4 and 5 working from left to right. If the symbol read from tape 4 is a morphism, the *TM* lets it act on the symbol read from tape 2 and if it is a reversal morphism, the *TM* lets it act on the symbol read from tape 3. Then *TM* writes $h(p)$ on tape 5. Simultaneously, *TM* compares $h(p)$ symbol by symbol with the contents of tape 1. If this comparison is successful, *TM* reads the next symbol from tapes 2, 3 and 4 and repeats the action described above, Otherwise, *TM* halts in a non-accepting state. If $h(p)$ is empty, the heads of tapes 1 and 5 do not move and tapes 2, 3 and 4 continue on to the next symbol. If tape 1 is exhausted but tapes 2, 3, 4 and 5 are not empty or if tape 1 is not exhausted when tapes 2, 3, 4 and 5 become empty, then *TM* halts in a non-accepting state. *TM* halts and accepts string y only if tapes 1, 2, 3, 4 and 5 become empty at the same time and all of the comparisons are successful. In this phase the time required is $k|y|$ at most.

During the entire computation each read–write head of *TM* makes at most one reversal in each phase. Therefore, *TM* is reversal-bounded. The total running time is at most

$$2|y|+k|y|+2k|y|+2k|y|+2k|y|+2k|y|+k|y| = (10k+2)|y|.$$

Since k is a constant, if t is chosen to be any integer greater then $10k+2$, then *TM* operates in linear time equal to $t|y|$. Hence, $L(TM)$ is in $\mathsf{L}_{\mathrm{BNP}}$ by Lemma 3.4. Clearly, $L(TM) = W(L)$ so that $W(L)$ is also in $\mathsf{L}_{\mathrm{BNP}}$. ■

Corollary 4.1. The class $\mathsf{L}_{\mathrm{BNP}}$ is the smallest class of languages containing the regular sets and closed under the intersection and the general morphic replication of a linear-erasing morphism.

Proof. Suppose L is the smallest class containing the regular sets and closed under the intersection and the generalized morphic replication of a linear-erasing morphism. By Lemma 3.1 every linear context-free language can be expressed by a regular set, a linear-erasing morphism and a linear-erasing reversal morphism. From Lemma 3.4, every language in $\mathsf{L}_{\mathrm{BNP}}$ can be expressed as the non-erasing morphic image of the intersection of three linear context-free languages, so that $\mathsf{L}_{\mathrm{BNP}} \subseteq \mathsf{L}$. By Theorem 4.1 $\mathsf{L}_{\mathrm{BNP}}$ is closed under the generalized morphic replication of a linear-erasing morphism. Obviously $\mathsf{L}_{\mathrm{BNP}}$ is closed under intersection and contains all regular sets, therefore, $\mathsf{L} \subseteq \mathsf{L}_{\mathrm{BNP}}$. ■

Theorem 4.2. The class *NP* is closed under the generalized morphic replication of a polynomial-erasing morphism.

The proof of Theorem 4.2 is similar to that of Theorem 4.1 and is omitted.

Corollary 4.2. The class *NP* is the smallest class of

languages containing the regular sets and closed under the intersection and the generalized morphic replication of a polynomial-erasing morphism.

Proof. Suppose L is the smallest class containing the regular sets and closed under the intersection and the generalized morphic replication of a polynomial-erasing morphism. Clearly, $L_{BNP} \subseteq NP$. In ref. 1 Baker showed that for $L_1 \in NP$ there exists $L_2 \in L_{BNP}$ and a polynomial-erasing morphism h such that $h(L_2) = L_1$. Therefore, $NP \subseteq L$. By Theorem 4.2 NP is closed under the generalized morphic replication of a polynomial erasing morphism. Obviously, the class NP contains regular sets and is closed under the intersection. Therefore, $L \subseteq NP$. ■

Theorem 4.3. The class of recursively enumerable sets is closed under the generalized morphic replication of an arbitrary morphism.

The proof of Theorem 4.3 is similar to that of Theorem 4.1 and is omitted.

Corollary 4.3. The class of recursively enumerable sets is the smallest class of languages containing the regular sets and closed under the intersection and the generalized morphic replication of an arbitrary morphism.

Corollary 4.3 can be proved by Lemma 3.1 and Lemma 3.2.

5. CONCLUDING REMARKS

In this paper is it shown that L_{BNP}, the class NP and the class of recursively enumerable sets are closed under the generalized morphic replication when the morphism is linear-erasing, polynomial-erasing, and arbitrary, respectively. Since the generalized morphic replication offers techniques to handle infinite sets of strings of morphisms and reversal morphisms, it is therefore, useful in decomposition of formal languages and automata.

The class of languages accepted in real time by nondeterministic reversal-bounded multi-tape Turing acceptors contains some context-sensitive languages such as $\{0^n 1^n 0^n \mid n > 0\}$. But the question remains whether the class L_{BNP} contains all context-sensitive languages.

Acknowledgement

We thank Professor R. Book for his comments and suggestions.

REFERENCES

1. B. Baker and R. Book, Reversal-bounded multi-pushdown machines. *J. Comptr Syst. Sci.* **8**, 315–332 (1974).
2. R. Book, M. Nivat and M. Paterson, Reversal-bounded acceptors and intersections of linear languages. *SIAM J. on Computing* **3**, 283–295 (1974).
3. R. Book, Simple representations of certain classes of language. *J. ACM* **1**, 23–31 (1978).
4. R. Book and S. Griebach, Quasi-real time languages. *Math. Systems Theory* **4**, 97–111 (1970).
5. S. Ginsburg and S. Griebach, Principal AFL. *J. Comptr Syst. Sci.* **4**, 308–338.
6. S. Ginsburg and E. Spanier, Finite-turn pushdown automata. *SIAM J. on Control* **4**, 429–453 (1966).
7. J. Hopcroft and J. Ullman, *Formal Languages and Their Relation to Automata.* Addison-Wesley, Reading, Mass. (1969).