JOURNAL OF COMPUTER AND SYSTEM SCIENCES 18, 18–34 (1979)

Some Decision Problems Concerning Sequential Transducers and Checking Automata*

EITAN M. GURARI AND OSCAR H. IBARRA

Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455

Received October 3, 1977; revised May 26, 1978

Boundary points between decidability and undecidability of various decision questions concerning two-way sequential transducers and checking automata are investigated. The results show how some unsolvable problems become solvable when certain restrictions (e.g., the machines being deterministic, reversal-bounded, etc.) are imposed. A solution to an open problem concerning cascade products of pushdown automata is also given.

INTRODUCTION

The simplest known machine which can map a language (= set of strings) into another is the deterministic one-way sequential transducer. The properties of one-way sequential transducers are well known. For example, it is a basic result that relational equivalence is decidable for deterministic sequential transducers but undecidable for the nondeterministic case [6]. The unsolvability of the latter holds even if the machines operate in real time and the input (or output) alphabet is restricted to one letter [10]. The set of output strings (i.e., transduction) defined by a nondeterministic transducer is regular [12]. Hence, all the usual decision questions concerning one-way sequential transductions are solvable.

A natural generalization of the one-way transducer is one which is allowed two-way motion on the input string. Clearly, two-way sequential transducers are much more powerful than one-way transducers (e.g., they can make duplicate copies of an input string). Two-way sequential transducers have been studied in several places. In [1] language-theoretic properties of two-way sequential transductions of regular sets and context-free languages are given, and in [13] a grammatical characterization of deterministic two-way sequential transductions of regular sets is shown. In a recent paper [8] (see also [5]), algebraic and grammatical characterizations of several restricted classes of two-way sequential tranductions of full semiAFL's are presented.

In this paper, we look at several decision questions concerning two-way sequential transducers. Section 1 gives some definitions and a few basic results. In particular, the relationship between two-way sequential transducers and checking automata [4] is established. Section 2 presents some undecidable properties of transductions defined by

* This research was supported by the National Science Foundation under Grant No. DCR75-17090.

0022-0000/79/010018-17$02.00/0
Copyright © 1979 by Academic Press, Inc.
All rights of reproduction in any form reserved.

two-way sequential transducers which make at most one input reversal. Section 3 investigates the decidable properties of transductions defined by two-way sequential transducers whose outputs are restricted to be in $w_1^* \cdots w_n^*$ ($n \geq 1$, each w_i a nonnull string).[1] Section 4 does a similar study for input restricted machines. The main result of the section shows that the containment, equivalence and disjointness problems are decidable for transductions defined by deterministic two-way transducers whose inputs come from $w_1^* \cdots w_n^*$. An open problem posed in [1], and which still remains open, is whether or not relational equivalence for deterministic two-way sequential transducers is decidable. Section 5 offers a positive answer for some restricted classes of deterministic two-way sequential transducers. Finally, Section 6 considers pushdown transducers and a related machine structure, the cascade product of pushdown automata, introduced recently in [15]. An open problem in [15] is generalized and a solution obtained.

1. Preliminaries

A *two-way finite-state sequential transducer* (abbreviated *FST*) is a two-way finite automaton with outputs [1, 8, 13]. (See Figure 1.) Thus a FST is an 8-tuple $M = \langle K, \Sigma, ¢, \$, \Delta, \delta, q_0, F\rangle$, where K, Σ, Δ, and F are finite nonempty sets of *states*, *input alphabet*, *output alphabet*, and *accepting states*, respectively. ¢ and \$ are *end markers for the input tape*, q_0 in K is the *start state*, and δ is a mapping from $K \times (\Sigma \cup \{¢, \$\})$ into

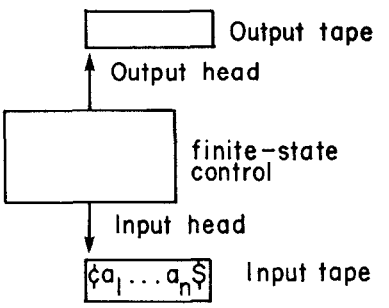

FIG. 1. A FST.

the finite subsets of $K \times \{-1, 0, +1\} \times \Delta^*$.[2] For (q, a) in $K \times (\Sigma \cup \{¢, \$\})$, if $\delta(q, a)$ contains (p, d, y) then M in state q scanning 'a' on the input tape may move its input head $d(= -1, 0, +1)$ places to the right, output string 'y' and enter state p. We assume that $d \geq 0$ if $a = ¢$ and $d \leq 0$ if $a = \$$. (Thus, the input head cannot fall off the input tape.) M is *deterministic* (*DFST*) if $|\delta(q, a)| \leq 1$[3] for each (q, a) in $K \times (\Sigma \cup \{¢, \$\})$ with $\delta(q, a) = \varnothing$ for each q in F. The *relation* defined by M is the set of input-output pairs $R(M) = \{(x, y) \mid x \text{ in } \Sigma^*, M \text{ when started in state } q_0 \text{ on the left end marker of } ¢x\$$

[1] $w_1^* \cdots w_n^*$ denotes the set $\{w_1^{i_1} \cdots w_n^{i_n} \mid i_1, \ldots, i_n \geq 0\}$.

[2] Δ^* denotes the set of all finite-length strings of symbols in Δ including the null string, ϵ.

[3] $|S|$ is the cardinality of S.

eventually enters an accepting state after producing output string y}. The *transduction* defined by M (or *language generated by* M) is the set $T(M) = \{ y \mid y \text{ in } \Delta^*, \text{ for some } x \text{ in } \Sigma^*, (x, y) \text{ is in } R(M)\}$.

FST's are computationally equivalent to checking automata studied in [4]. A *checking automaton* (CA) (see Figure 2) is an 8-tuple $M = \langle K, \Sigma, ¢, \$, \Delta, \delta, q_0, F\rangle$, where K, Σ, Δ, and F are finite nonempty sets of *states*, *checking tape alphabet*, *output alphabet*, and *accepting states*, respectively. ¢ and \$ are *end markers for the checking tape*, q_0 in K is the *start state*, and δ is a mapping from $K \times (\Sigma \cup \{¢, \$\}) \times (\Delta \cup \{\epsilon\})$ into the subsets of $K \times \{-1, 0, +1\}$. For (q, a, b) in $K \times (\Sigma \cup \{¢, \$\}) \times (\Delta \cup \{\epsilon\})$, if $\delta(q, a, b)$ contains (p, d) then M in state q scanning 'a' on the checking tape and 'b' on the output tape may move its output head to the right of b (if $b \neq \epsilon$), move its checking head d places to the right, and enter state p. Again we assume that $d \geqslant 0$ if $a = ¢$ and $d \leqslant 0$ if $a = \$$. M is *deterministic* (DCA) if $|\delta(q, a, b)| \leqslant 1$ for all (q, a, b) in $K \times (\Sigma \cup \{¢, \$\}) \times (\Delta \cup \{\epsilon\})$, and either $\delta(q, a, \epsilon) = \varnothing$ or $\delta(q, a, b) = \varnothing$ for all b in Δ. The *language accepted* by M is the set $T(M) = \{ y \mid y \text{ in } \Delta^*$, there exists a checking tape $¢x\$$, x in Σ^*, such that M when started in state q_0 with its output head on the left end of y and its checking head on the left end marker of $¢x\$$ eventually reads all of y and lands in an accepting state}. Note that if q_0 is an accepting state then ϵ is in $T(M)$.

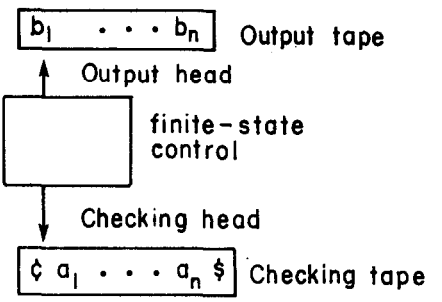

FIG. 2. A CA.

Remark. For notational consistency, we have referred to Δ in the specification of a CA as the output alphabet. In the original formulation [4], Δ is referred to as the input alphabet.

A FST(CA)M is *k-reversal* if the number of times M reverses on any input tape (checking tape) in an accepting computation is at most k. We use the notation k-FST, k-CA, k-DFST, k-DCA for the k-reversal machines. Note that a 0-FST (0-DFST) is equivalent to a one-way nondeterministic (deterministic) sequential transducer.

If $M = \langle K, \Sigma, ¢, \$, \Delta, \delta, q_0, F\rangle$ is a FST, DFST, CA, etc., then we say that M is over $\Sigma^* \times \Delta^*$. If the output strings are known to take a special form e.g., if $T(M) \subseteq w_1^* \cdots w_n^*$ ($n \geqslant 1$, each w_i in Δ^+),[4] then we say that M is over $\Sigma^* \times w_1^* \cdots w_n^*$. The same convention is used when the inputs (or checking tapes) take a special form. Thus a DFST M over $w_1^* \cdots w_n^* \times \Delta^*$ is such that $\{x \mid (x, y) \text{ is in } R(M) \text{ for some } y\} \subseteq w_1^* \cdots w_n^*$. Similarly, a

[4] $\Delta^+ = \Delta^* - \{\epsilon\}$.

k-DCA M over $0^*1^* \times \Delta^*$ has the property that the checking tape for each accepting computation is of the form $\not{c}x\$$, x in 0^*1^*, and the checking head reverses at most k times.

The connection between FST's and CA's is given by the following proposition.

PROPOSITION 1.1. *Let $L \subseteq \Delta^*$. Then $L = T(M_1)$ for some* FST M_1 *over $\Sigma^* \times \Delta^*$ if and only if $L = T(M_2)$ for some* CA M_2 *over $\Sigma^* \times \Delta^*$.*

Proof. The construction from one machine to the other is straightforward (see, e.g., [13]). ∎

COROLLARY 1.1. *Transductions defined by* 0-FST's (= *languages accepted by* 0-CA's) *are regular* [12]. *Thus all standard decision questions concerning transductions of* 0-FST's *are decidable.*

The next proposition follows from Proposition 1.1 and the decidability of similar problems for one-way stack automata [2].

PROPOSITION 1.2. *It is decidable to determine for a given* FST (CA) M *whether $T(M)$ is empty, finite or infinite.*

We conclude this section with a theorem relating the reversal-bounded classes.

THEOREM 1.1. *The following statements are equivalent for any set $L \subseteq \Delta^*$ and $k \geqslant 0$*

(1) $L = T(M_1)$ *for some* k-CA M_1,

(2) $L = T(M_2)$ *for some* k-FST M_2,

(3) $L = T(M_3)$ *for some* k-DFST M_3,

(4) $L = T(M_4)$ *for some* k-DCA M_4,

(5) *For some regular set R and homomorphisms $h_1, \ldots, h_{k+1}$, $L = \{h_1(x)\, h_2(x^R) \cdots h_{k+1}(x^R) \mid x \text{ in } R\}$ if k is odd or $L = \{h_1(x)\, h_2(x^R) \cdots h_k(x^R)\, h_{k+1}(x) \mid x \text{ in } R\}$ if k is even, where x^R = reverse of x.*

Moreover, for the equivalences (1) *and* (2), *M_1 is over $\Sigma^* \times \Delta^*$ for some Σ if and only if M_2 is over $\Sigma^* \times \Delta^*$.*

Proof. That (1) implies (2), (3) implies (4), and (4) implies (1) are obvious. The construction of M_3 given M_2 uses the technique described in [5]. Intuitively, an input to M_3 has $k + 2$ 'tracks'. Track 1 contains the original input to M_2 while track $i + 1$ contains the 'encoding' of possible moves of M_2 on the ith pass over the input. Details can be found in [5]. The proof of the equivalence of (1) and (5) can also be found in [5]. ∎

2. UNDECIDABLE PROPERTIES OF 1-REVERSAL MACHINES

In this section we show the undecidability of the universe problem for 1-DCA's and 1-FST's over $0^*1^* \times \Delta^*$. This result does not hold for 1-DFST's since we can show

(see Section 4) that DFST's over $w_1^* \cdots w_n^* \times \Delta^*$ have a solvable containment problem. We also discuss briefly the status of the disjointness problem for these machines.

THEOREM 2.1. *The universe, containment and equivalence problems for* 1-DCA's *over* $0^*1^* \times \Delta^*$ *are undecidable.*

Proof. It is sufficient to prove the undecidability of the universe problem. Let Z be a single-tape Turing machine with state set K and tape alphabet $\Gamma = \{0, 1, b\}$ (b for blank). Assume that $K \cap \Gamma = \varnothing$, and let $\Delta = K \cup \Gamma \cup \{\#\}$, where $\#$ is a new symbol. We may also assume that the start state q_0 of Z is not a halting state and Z does not write blanks. Then any configuration of Z can be represented by a string of the form xqy, where q is in K and xy is in Γ^*. The initial configuration corresponding to the blank tape is q_0b. Define the language $L = \{x \mid x \text{ in } \Delta^*, x \text{ is not of the form } \alpha_1\#\alpha_2\# \cdots \#\alpha_k \text{ where } k \geqslant 2, \alpha_1, \ldots, \alpha_k \text{ are configurations of } Z, \alpha_1 \text{ is the initial configuration, } \alpha_k \text{ is a halting configuration, and for each } 1 \leqslant i < k, \alpha_{i+1} \text{ is a proper successor of } \alpha_i\}$. Clearly, $L = \Delta^*$ if and only if Z does not halt. A 1-DCA M over $0^*1^* \times \Delta^*$ can be constructed such that $T(M) = L$. The construction is standard: Given output string y and checking tape $\not\in 0^i1^j\$$, M uses 0^i1^j to check that α_{i+1} is not a proper successor of α_i with the error occuring in position j, $j+1$ or $j+2$ of α_i and α_{i+1}. (Note that strings which are not of the form $\alpha_1\# \cdots \#\alpha_k$ for other reasons can be easily checked by the finite state control). The undecidability of the universe problem now follows from the unsolvability of the halting problem for Turing machines [7]. ∎

We shall show in Section 3 that the containment, equivalence and disjointness problems are decidable for k-CA's over $\Sigma^* \times w_1^* \cdots w_n^*$. Thus Theorem 2.1 is not symmetric with respect to input/output.

The next result is a direct consequence of Theorems 1.1 and 2.1.

THEOREM 2.2. *The universe, containment and equivalence problems for* 1-FST's *over* $0^*1^* \times \Delta^*$ *are undecidable.*

Theorem 2.2 does not hold for 1-DFST's over $0^*1^* \times \Delta^*$. In fact, we shall show in Section 4 that these problems are decidable for DFST's over $w_1^* \cdots w_n^* \times \Delta^*$, where $n \geqslant 1$, each w_i in Σ^+. However, from Theorems 1.1 and 2.2, we have the following result.

COROLLARY 2.1. *The universe, containment and equivalence problems for* 1-DFST's *over* $\Sigma^* \times \Delta^*$ ($|\Sigma| \geqslant 2$) *are undecidable.*

The next result which was shown in [1] concerns the disjointness problem. For completeness, we outline the proof in [1].

THEOREM 2.3. *It is undecidable to determine for* 1-DFST's M_1 *and* M_2 *whether* $T(M_1) \cap T(M_2) = \varnothing$.

Proof. Let (X, Y) be an instance of the Post correspondence problem (*PCP*) [7], where $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n)$, each x_i, y_i in $\{0, 1\}^+$. Clearly, we can construct

1-DFST's M_1 and M_2 such that $R(M_1) = \{(10^{i_1} \cdots 10^{i_k}, 10^{i_1} \cdots 10^{i_k} \# x_{i_k}^R \cdots x_{i_1}^R) \mid k \geqslant 1, 1 \leqslant i_j \leqslant n\}$ and $R(M_2) = \{(10^{i_1} \cdots 10^{i_k}, 10^{i_1} \cdots 10^{i_k} \# y_{i_k}^R \cdots y_{i_1}^R) \mid k \geqslant 1, 1 \leqslant i_j \leqslant n\}$. Then $T(M_1) \cap T(M_2) \neq \varnothing$ if and only if there exist $i_1, \ldots, i_k$ such that $x_{i_k}^R \cdots x_{i_1}^R = y_{i_k}^R \cdots y_{i_1}^R$, i.e., if and only if (X, Y) has a solution. The result follows since the PCP is undecidable [7]. ∎

For FST's, we can also prove a somewhat stronger result.

PROPOSITION 2.1. *It is undecidable to determine for* FST's M_1 *and* M_2 *over* $1^* \times \Delta^*$ *whether* $T(M_1) \cap T(M_2) = \varnothing$.

Proof. This follows from the unsolvability of the disjointness problem for deterministic one-way one-counter machines [9, 14] and the observation that given any deterministic (nondeterministic) one-way one-counter machine M, we can effectively construct a FST M' over $1^* \times \Delta^*$ such that $T(M') =$ the set accepted by M. ∎

Theorem 2.3 and Propositions 2.1 are best possible since we shall show later that the disjointness problem for DFST's (or for k-FST's) over $w_1^* \cdots w_n^* \times \Delta^*$ ($n \geqslant 1$, each w_i in Σ^+) is dedidable.

3. DECIDABLE PROPERTIES OF OUTPUT RESTRICTED MACHINES

In this section, we look at DFST's (k-FST's) over $\Sigma^* \times w_1^* \cdots w_n^*$, $n \geqslant 1$, each w_i in Δ^+. We prove that the containment, equivalence and disjointness problems for transductions defined by these machines are decidable. The proof uses a result in [13] connecting DFST's to absolutely parallel grammars.

DEFINITION. An *absolutely parallel grammar* (*APG*) [13] is a 4-tuple $G = \langle N, \Delta, S, P \rangle$, where N and Δ are finite nonempty disjoint sets of *nonterminals* and *terminals*, respectively, S in N is the *start symbol*, and P is a finite set of *productions* π of the form $(A_1, \ldots, A_k) \to (y_1, \ldots, y_k)$ with $k \geqslant 1$, each A_i in N and y_i in $(N \cup \Delta)^*$.

For x and y in $(N \cup \Delta)^*$, we write $x \Rightarrow y$ if for some $\pi\colon (A_1, \ldots, A_k) \to (y_1, \ldots, y_k)$, $x = x_1 A_1 x_2 A_2 \cdots x_k A_k x_{k+1}$, $y = x_1 y_1 x_2 \cdots x_k y_k x_{k+1}$ and $x_1, \ldots, x_{k+1}$ are in Δ^*. The reflexive-transitive closure of $\Rightarrow$ is denoted by $\overset{*}{\Rightarrow}$. The language generated by G is $L(G) = \{x \mid x \text{ in } \Delta^*, S \overset{*}{\Rightarrow} x\}$.

The following theorem is the main result of [13].

THEOREM 3.1. *The following statements are equivalent for* $L \subseteq \Delta^*$:

(1) $L = T(M)$ *for some* DFST M *over* $\Sigma^* \times \Delta^*$.

(2) $L = L(G)$ *for some* APG G.

Moreover, one can effectively construct G *from* M, *and conversely.*

Next we recall the definition of semilinear sets and the Parikh mapping of a language [3, 11].

DEFINITION. Let N be the set of natural numbers. A subset $Q \subseteq N^n$ is a *linear set* if there exist n-tuples of natural numbers $v_0, v_1, \ldots, v_r$ such that $Q = \{x \mid x = v_0 + c_1v_1 + \cdots + c_rv_r$, each c_i in $N\}$. v_0 is the *constant* and $v_1, \ldots, v_r$ are the *periods* of Q. Any finite union of linear sets is called a *semilinear set*. A semilinear set Q is uniquely specified by giving the constant and periods of each linear set of Q.

Let $\Delta = \{a_1, \ldots, a_n\}$. If x is in Δ^*, let $f(x) = (\#a_1(x), \ldots, \#a_n(x))$, where $\#a_i(x) =$ number of occurrences of symbol a_i in x. Note that $f(\epsilon) = (0, \ldots, 0)$. If $L \subseteq \Delta^*$, let $f(L) = \{f(x) \mid x \text{ in } L\}$. $f(L)$ is called the *Parikh mapping* of L.

The next lemma says that $f(L(G))$ is an effectively computable semilinear set for any APG G. This result was recently shown in [5]. For completeness, we give a short proof which is essentially the one given in [5].

LEMMA 3.1. *Let $G = \langle N, \Delta, S, P\rangle$ be an* APG. *Then $f(L(G))$ is an effectively computable semilinear set.*

Proof. We construct from G a right-linear grammar [7] $G' = \langle N', \Delta, S', P'\rangle$, where $N' = \{\langle A_1, \ldots, A_k\rangle, \langle B_1, \ldots, B_m\rangle \mid A_1, \ldots, A_k, B_1, \ldots, B_m \text{ in } N, (A_1, \ldots, A_k) \rightarrow (y_1, \ldots, y_k)$ in P, $y_1 \cdots y_k = x_1B_1x_2 \cdots x_mB_mx_{m+1}$ for some $x_1, \ldots, x_{m+1}$ in $\Delta^*\}$,[5] $S' = \langle S\rangle$, and $P' = \{\langle A_1, \ldots, A_k\rangle \rightarrow x_1 \cdots x_{m+1}\langle B_1, \ldots, B_m\rangle \mid (A_1, \ldots, A_k) \rightarrow (y_1, \ldots, y_k)$ in P, $y_1 \cdots y_k = x_1B_1x_2 \cdots x_mB_mx_{m+1}\} \cup \{\langle A_1, \ldots, A_k\rangle \rightarrow x_1 \cdots x_k \mid \langle A_1, \ldots, A_k\rangle \rightarrow (x_1, \ldots, x_k)$ in P, each x_i in $\Delta^*\}$. Since G' is a right-linear grammar, $f(L(G'))$ is an effectively computable semilinear set [3, 11]. The result follows since $f(L(G)) = f(L(G'))$. ∎

COROLLARY 3.1. *Let $G = \langle N, \Delta, S, P\rangle$ be an* APG *and $w_1, \ldots, w_n$ be in Δ^+ such that $L(G) \subseteq w_1^* \cdots w_n^*$. Then the set $Q = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n}$ in $L(G)\}$ is an effectively computable semilinear set.*

Proof. Let $a_1, \ldots, a_n$ be distinct symbols, and define a homomorphism h by: $h(a_i) = w_i$ for each $1 \leqslant i \leqslant n$. Clearly, $L' = h^{-1}(L(G)) \cap a_1^* \cdots a_n^* = \{a_1^{i_1} \cdots a_n^{i_n} \mid w_1^{i_1} \cdots w_n^{i_n}$ in $L(G)\}$. Since the class of APG's is effectively closed under AFL operations [1, 13] we can effectively construct an APG G' such that $L(G') = L'$. By Lemma 3.1, $Q = f(F(G'))$ is an effectively computable semilinear set. ∎

The containment, equivalence and disjointness problems for semilinear sets are decidable [3]. The next theorem then follows from Theorem 3.1, Corollary 3.1 and Theorem 1.1.

THEOREM 3.2. *The containment, equivalence and disjointness problems are decidable for each of the following families of machines*:

(1) DFST's *over* $\Sigma^* \times w_1^* \cdots w_n^*$,

(2) k-FST's *over* $\Sigma^* \times w_1^* \cdots w_n^*$,

(3) k-CA's *over* $\Sigma^* \times w_1^* \cdots w_n^*$.

Remark. We are unable to strengthen Theorem 3.2 (2) and (3) to include FST's

[5] $\langle A_1, \ldots, A_k\rangle$, $\langle B_1, \ldots, B_m\rangle$ are abstract symbols.

and CA's respectively. We should point out, however, that if M is a FST (or a CA), $f(T(M))$ need not be semilinear. For example, $L = \{a^n b^{kn} \mid k, n \geqslant 1\}$ can be accepted by a FST (or by a CA), but $f(L)$ is not semilinear.

4. Decidable Properties of Input Restricted Machines

In this section we study DFST's over $w_1^* \cdots w_n^* \times \Delta^*$, i.e., DFST's whose inputs come from a bounded language. It may seem that since the number of reversals is unbounded, the input tape can be used as a counter and one might suspect that the containment problem for these machines is undecidable. (The containment problem for deterministic one-way one-counter machines is undecidable [9, 14].) However, we show that the containment, equivalence and disjointness problems for transductions defined DFST's over $w_1^* \cdots w_n^* \times \Delta^*$ are solvable.

We begin with the following lemma.

Lemma 4.1. *Let M be a* DFST(FST) *over $w_1^* \cdots w_n^* \times \Delta^*$ and $a_1, \ldots, a_n$ be distinct symbols. We can effectively construct a* DFST(FST) *M' over $a_1^* \cdots a_n^* \times \Delta^*$ such that $R(M') = \{(a_1^{i_1} \cdots a_n^{i_n}, y) \mid (w_1^{i_1} \cdots w_n^{i_n}, y)$ is in $R(M)\}$. Moreover, if M is k-reversal then so is M'.*

Proof. M' simulates the computation of M on w_j in the states. ∎

Lemma 4.2. *Let M be a* DFST *over $a_1^* \cdots a_n^* \times \Delta^*$. We can effectively construct a* DFST *M' over $a_1^* \cdots a_n^* \times \Delta^*$ such that $R(M') = R(M)$ and M' on input $\mathrm{¢} a_1^{i_1} \cdots a_n^{i_n} \$$ only makes reversals on the boundaries between the symbols ¢, $a_1, \ldots, a_n$, \$.*

Proof. M' has a buffer B which can hold a string of length s, where s is the number of states of M. Initially, B contains ϵ. It will be seen that the string in B will always be of the form a_j^l, $0 \leqslant l \leqslant s$, and that B is reset to ϵ everytime a new input segment $a_j^{i_j}$ is entered. We describe the operation of M' on an input segment $a_j^{i_j}$. Assume that M' is entering the segment from the left. (The case when the segment is entered from the right is treated similarly.)

(1) M' simulates the computation of M. Whenever the input head is moved one position to the right, the string in B is shifted by one position to the left and the symbol which was under the input head just before the move is stored in the rightmost position of B.

(2) Suppose the buffer becomes full. In this case, since M is deterministic, it can no longer make a reversal until it reaches the right boundary of $a_j^{i_j}$. Thus, in this case, M' resets B to ϵ and continues the simulation of M on $a_j^{i_j}$ without using the buffer.

(3) Suppose M' reaches the right boundary of $a_j^{i_j}$ before or at the same time the buffer becomes full. In this case M' just resets the buffer to ϵ.

(4) Suppose M' is not on the right boundary of $a_j^{i_j}$ and it wants to make a reversal, but the buffer is not full. In this case, M' does not move its input head but simulates the

computation of M on the string stored in the buffer, starting on the symbol which corresponds to the symbol under the input head of M. Two cases arise:

Case 1. In the simulation M tries to leave the buffer from the right. In this case, M' goes on with the simulation as described in (1).

Case 2. In the simulation M tries to leave the buffer from the left. In this case, M' resets the buffer to ϵ and moves its input head to the right boundary of $a_j^{i_j}$ and then back to the left boundary of $a_j^{i_j}$.

(1)–(4) above show how a computation on segment $a_j^{i_j}$ can be simulated by M' without its input head reversing on the segment. It follows that M' can be constructed so that $R(M') = R(M)$ and M' on input $\cent a_1^{i_1} \cdots a_n^{i_n}\$$ only makes reversals on the boundaries between the symbols $\cent, a_1, \ldots, a_n, \$$. ∎

We can use Lemma 4.2 to convert a DFST over $a^* \cdots a^* \times \Delta^*$ to an equivalent k-DFST M'.

LEMMA 4.3. *Let M be a* DFST *over $a_1^* \cdots a_n^* \times \Delta^*$. We can effectively construct a k-*DFST *M' such that*

(1) $R(M') = R(M)$;

(2) *M' is full-sweep, i.e., it only makes reversals on the end markers*;

(3) *M' is nonstopping, i.e., it has no stationary moves on the input.*

Proof. Given M, we can construct a DFST M_1 satisfying Lemma 4.2, i.e., $R(M_1) = R(M)$ and M_1 only makes reversals on the boundaries between symbols $\cent, a_1, \ldots, a_n . \$$. Clearly, since the inputs are of the form $\cent a_1^{i_1} \cdots a_n^{i_n}\$$ (with n fixed), we can effectively construct another DFST M_2 such that $R(M_2) = R(M_1)$ and M_2 is full-sweep, i.e., it only makes reversals on the end markers. Now let s be the number of states of M_2. Consider any input $\cent x\$ = \cent a_1^{i_1} \cdots a_n^{i_n}\$$. If M_2 eventually halts on $\cent x\$$, then M_2 can visit an end marker at most s times. Moreover, M_2 can have at most $s - 1$ stationary moves on any input symbol. It follows that M_2 can be modified to an equivalent full-sweep, nonstopping k-DFST M', where $k = 2(s - 1)$. ∎

The next lemma shows that $T(M)$ for a DFST M over $a_1^* \cdots a_n^* \times \Delta^*$ is a bounded language.

LEMMA 4.4. *Let M be a* DFST *over $a_1^* \cdots a_n^* \times \Delta^*$. We can effectively find $w_1, \ldots, w_r$ such that $T(M) \subseteq w_1^* \cdots w_r^*$.*

Proof. By Lemma 4.3 we may assume that M is a full-sweep, nonstopping k-DFST. Let $M = \langle K, \Sigma, \cent, \$, \Delta, \delta, q_0, F\rangle$. Define functions σ and λ as follows: Let q be in K and x be in $(\Sigma \cup \{\cent\})^+$. If M when started in state q on the leftmost symbol of x makes a nonstopping sweep of x, outputs y and enters state p, then let $\sigma(q, x) = p$ and $\lambda(q, x) = y$. Otherwise, $\sigma(q, x)$ and $\lambda(q, x)$ are undefined. Also define $\sigma(q, \epsilon) = q$ and $\lambda(q, \epsilon) = \epsilon$.

(a) For q in K and $1 \leqslant i \leqslant n$, let $H(q, a_i) = \{ y \mid \text{for some } j, \lambda(q, a_i^j) = y\}$. We

claim that $H(q, a_i) \subseteq u_1^* \cdots u_s^* v_1^* \cdots v_t^*$ and $u_1, \ldots, u_s, v_1, \ldots, v_t$ in Δ^+ can effectively be found.

Since the number of states of M is finite, we can effectively find the least integer $s \geq 0$ (if it exists) such that for some $t \geq 1$ and p in K, $\sigma(q, a_i^s) = \sigma(q, a_i^{s+t}) = p$. Let $u_1 = \lambda(q, a_i^1), \ldots, u_s = \lambda(q, a_i^s)$, $v_1 = \lambda(p, a_i^t)$, $v_2 = \lambda(p, a_i^1), \ldots, v_t = \lambda(p, a_i^{t-1})$.[6] Clearly, $H(q, a_i) \subseteq u_1^* \cdots u_s^* v_1^* \cdots v_t^*$.

(b) Now for q in K, define $L(q) = \{y \mid \text{for some } a_1^{i_1} \cdots a_n^{i_n}, \lambda(q, \not{c} a_1^{i_1} \cdots a_n^{i_n}) = y\}$ and $R(q) = \{y \mid \text{for some } a_n^{i_n} \cdots a_1^{i_1}, \lambda'(q, \$a_n^{i_n} \cdots a_1^{i_1}) = y\}$.[7] From (a), since the number of states is finite and n is fixed, we can effectively find $z_1, \ldots, z_m$ in Δ^+ such that $L(q) \subseteq z_1^* \cdots z_m^*$ and $R(q) \subseteq z_1^* \cdots z_m^*$.

(c) From (b) and the fact that M is a full-sweep nonstopping k-DFST, we can effectively find $w_1, \ldots, w_r$ in Δ^+ such that $T(M) \subseteq w_1^* \cdots w_r^*$. ∎

The next lemma combines Lemmas 4.1 to 4.4.

LEMMA 4.5. *Let M be a* DFST *over $w_1^* \cdots w_n^* \times \Delta^*$ and $a_1, \ldots, a_n$ be distinct symbols. We can effectively find $z_1, \ldots, z_r$ in Δ^+ and a k-*DFST *M' over $a_1^* \cdots a_n^* \times z_1^* \cdots z_r^*$ such that $R(M') = \{(a_{i_1} \cdots a_{i_n}, y) \mid (w_{i_1} \cdots w_{i_n}, y)$ is in $R(M)\}$. (Thus, $T(M) = T(M')$ is a bounded language.)*

From Theorem 3.2 and Lemma 4.5 we have the main result of this section.

THEOREM 4.1. *The containment, equivalence and disjointness problems for* DFST*'s over $w_1^* \cdots w_n^* \times \Delta^*$ are decidable.*

Theorem 4.1 cannot be extended to FST's over $w_1^* \cdots w_n^* \times \Delta^*$ since by Theorem 2.2, the universe, containment and equivalence problems for 1-FST's over $0^*1^* \times \Delta^*$ are undecidable. However, we can show that the disjointness problem for k-FST's over $w_1^* \cdots w_n^* \times \Delta^*$ is decidable. In order to prove this, we need a result concerning multi-counter machines [9].

Notation. Let $DC(l, m, r)[NC(l, m, r)]$ denote the class of deterministic [nondeterministic] two-way finite automata augmented by l counters. Each machine in the class operates in such a way that in every accepting computation the count in each counter alternately increases and decreases at most m times and the input head reverses direction at most r times.

The following theorem summarizes the results about counter machines [9] that we will need in the paper.

THEOREM 4.2. (a) *The disjointness problem for the class $NC(l, m, r)$ is decidable.*

(b) *The disjointness, containment and equivalence problems for the class $DC(l, m, r)$ are decidable.*

[6] If u_i (or v_i) $= \epsilon$, replace it by some symbol b in Δ. (This way, the u_i's and v_i's are nonnull.)

[7] λ' is a function similar to λ except that it handles right to left nonstopping sweep.

(c) *The disjointness, containment and equivalence problems for machines in NC(l, m, r) restricted to inputs over $w_1^* \cdots w_n^*$ are decidable.*

Using Theorem 4.2, we can now prove

THEOREM 4.3. *The disjointness problem for k-FST's over $w_1^* \cdots w_n^* \times \Delta^*$ is decidable.*

Proof. By Theorem 4.2(a), it is sufficient to give an effective procedure for constructing for a given FST M over $w_1^* \cdots w_n^* \times \Delta^*$ a machine M' in $NC(l, m, r)$, for some l, m and r, such that $T(M') = T(M)$. By Lemma 4.1, we may assume that $w_i = a_i$, each a_i a distinct symbol, $1 \leqslant i \leqslant n$. We describe the operation of M' briefly.

Given input y, M' nondeterministically stores in n counters integers $i_1, \ldots, i_n$. Then M' simulates the computation of M on input $\mathfrak{c}a_1^{i_1} \cdots a_n^{i_n}\$$, and verifies that the output is y. Since M is reversal-bounded, M can only make a finite number of reversals on each segment $a_j^{i_j}$. Thus, the simulation can be carried out by M' by using a finite number of additional counters and making duplicate copies of integers $i_1, \ldots, i_n$. We omit the details. ∎

5. RELATIONAL EQUIVALENCE

So far we have only looked at decision questions concerning transductions defined by DFST's (FST's). In this section, we briefly investigate the relational equivalence problem. It is a well known elementary result in automata theory that relational equivalence for 0-DFST's (i.e., deterministic one-way transducers) is decidable. However, the relational equivalence problem for 0-FST's is undecidable [6]. In fact, in a recent paper [10], it is shown that relational equivalence for 0-FST's over $1^* \times \Delta^*$ (or over $\Sigma^* \times 1^*$) is undecidable, even if the machines are nonstopping and all states are accepting. Thus, for nondeterministic transducers there is nothing to do.

For the deterministic case, it is an open question [1] whether the relational equivalence problem is decidable for DFST's. We have no solution to this problem, but we offer a positive answer in the case when the DFST's are over $w_1^* \cdots w_n^* \times \Delta^*$.

THEOREM 5.1. *The following problems are decidable for* DFST's *M_1 and M_2 over $w_1^* \cdots w_n^* \times \Delta^*$:*

(1) *Is $R(M_1) \subseteq R(M_2)$?*

(2) *Is $R(M_1) = R(M_2)$?*

Proof. By Lemma 4.1, we may assume that $w_i = a_i$, each a_i a distinct symbol, $1 \leqslant i \leqslant n$. We may also assume by Lemma 4.3 that each M_i is a full-sweep, nonstopping k-DFST. Let $\#$ be a new symbol, and define the language $L(M_i) = \{a_1^{i_1} \cdots a_n^{i_n} \# y \mid (a_1^{i_1} \cdots a_n^{i_n}, y)$ in $R(M_i)\}$, $i = 1, 2$. Clearly, the language $L(M_i)$ can be accepted by a deterministic multicounter machine M_i' in $DC(l, m, r)$ for some l, m, r. The result now follows from Theorem 4.2(b). ∎

We have not been able to obtain an analog of Theorem 5.1 for DFST's over $\Sigma^* \times w_1^* \cdots w_n^*$. However, we can prove the following weaker result.

THEOREM 5.2. *The following problems are decidable for* k-DFST*'s over* $\Sigma^* \times w_1^* \cdots w_n^*$:

(1) *Is* $R(M_1) \subseteq R(M_2)$?

(2) *Is* $R(M_1) = R(M_2)$?

Proof. Let $a_1, \ldots, a_n, \#$ be distinct symbols. For $i = 1, 2$, let $L(M_i) = \{a_1^{i_1} \cdots a_n^{i_n} \# x \mid (x, w_1^{i_1} \cdots w_n^{i_n})$ in $R(M_i)\}$. We shall construct a deterministic multicounter machine M_i' in $DC(l, m, r)$ for some l, m, r such that $T(M_i') = L(M_i)$. By Theorem 4.2(b), the result would follow. We describe the construction of M_1'. The construction of M_2' is similar.

M_1' has a buffer of size $s = \max\{|\, w_1 \,|, \ldots, |\, w_n \,|\}$ which is initially empty. M_1' on input $\not{c} a_1^{i_1} \cdots a_n^{i_n} \# x \$$ makes a pass on $a_1^{i_1} \cdots a_n^{i_n}$ and stores the integers $i_1, \ldots, i_n$ in n counters, $c_1, \ldots, c_n$. Assume without loss of generality that $i_j \geqslant 1$ for $1 \leqslant j \leqslant n$. Let $j = 1$. M_1' simulates the computation of M_1 on input x, storing the output in the buffer. When the buffer contains $w_j z$ for some z, M' then decrements c_j by 1, deletes w_j from the buffer, and left justifies z in the buffer. M_1' iterates the process of generating a string in the buffer, decrementing c_j, deleting w_j, and shifting the remaining string until c_j becomes zero. When this happens, j is incremented by 1 and the computation described is repeated for this new value of j. M_1' accepts the input if during the simulation M_1 enters an accepting state and the values of counters $c_1, \ldots, c_n$ are zero. ∎

If M_1 and M_2 in the proof of Theorem 5.2 are k-FST's over $z_1^* \cdots z_r^* \times w_1^* \cdots w_n^*$ then $L(M_i) \subseteq a_1^* \cdots a_r^* \# z_1^* \cdots z_n^*$. The following corollary then follows from Theorem 4.2(c).

COROLLARY 5.1. *The following problems are decidable for* k-FST*'s over* $z_1^* \cdots z_r^* \times w_1^* \cdots w_s^*$:

(1) *Is* $R(M_1) \subseteq R(M_2)$?

(2) *Is* $R(M_1) = R(M_2)$?

Remark. As we have mentioned earlier, the relational equivalence problem for 0-FST's over $1^* \times \Delta^*$ (or over $\Sigma^* \times 1^*$) is undecidable. Hence Corollary 5.1 is best possible.

Finally we mention a result concerning the disjointness problem for relations.

COROLLARY 5.2. (a) *It is decidable to determine for* k-FST*'s* M_1 *and* M_2 *over* $\Sigma^* \times w_1^* \cdots w_n^*$ *(or over* $w_1^* \cdots w_n^* \times \Delta^*$*) whether* $R(M_1) \cap R(M_2) = \varnothing$. (b) *It is decidable to determine for* DFST*'s* M_1 *and* M_2 *over* $w_1^* \cdots w_n^* \times \Delta^*$ *whether* $R(M_1) \cap R(M_2) = \varnothing$.

Proof. (a) follows from Theorem 4.2(a) and the constructions in the proof of Theorem 5.2. (b) follows from (a) and Lemma 4.5. ∎

6. Cascade Product of Pushdown Automata

We can equip a two-way sequential transducer with a pushdown store and obtain a two-way pushdown transducer. The undecidable decision problems for finite-state transducers will obviously remain unsolvable for pushdown transducers. There are other undecidable properties which we shall not discuss here since they can easily be shown using well known properties of context-free languages (e.g., the characterization of recursively enumerable sets as homomorphisms of the intersections of deterministic context-free languages [2].)

A machine structure which is somewhat related to a pushdown transducer is the cascade product of pushdown automata introduced recently in [15]. We briefly define the notion of a cascade product and then prove a result, a corollary of which, solves an open problem in [15].

Let $k \geqslant 2$. For $1 \leqslant i \leqslant k$, let $M_i = \langle K_i, \Sigma_i, \Gamma_i, \delta_i, q_{0i}, Z_{0i}, F_i \rangle$ be (one-way) pushdown automata (*PDA*) [7]. Thus, $K_i, \Sigma_i, \Gamma_i, F_i \subseteq K_i$ are finite nonempty sets of states, input alphabet, pushdown alphabet, and accepting states, respectively. q_{0i} is the start state, Z_{0i} is the initial pushdown symbol, and δ_i is a mapping from $K_i \times (\Sigma_i \cup \{\epsilon\}) \times \Gamma_i$ into the finite subsets of $K_i \times \Gamma_i^*$. M_i is *deterministic* (*DPDA*) if (1) $| \delta_i(q, a, Z)| \leqslant 1$ for each (q, a, Z) in $K_i \times (\Sigma_i \cup \{\epsilon\}) \times \Gamma_i$ and (2) for each (q, Z) in $K_i \times \Gamma_i$, either $\delta_i(q, \epsilon, Z) = \varnothing$ or $\delta(q, a, Z) = \varnothing$ for all a in Σ_i.

For $1 \leqslant i \leqslant k-1$, let ϕ_i be a mapping from $K_i \times (\Sigma_i \cup \{\epsilon\}) \times \Gamma_i$ into $\Sigma_{i+1} \cup \{\epsilon\}$. ($\phi_i$ is called a *cascade mapping*). The cascade product of $M_1, \ldots, M_k$ with respect to $\phi_1, \ldots, \phi_{k-1}$, denoted by $M_1\phi_1M_2\phi_2 \cdots \phi_{k-1}M_k$ is a one-way automaton with k pushdown stores (see Figure 3) whose specification is $\langle K_1 \times \cdots \times K_k, \Sigma_1, \Gamma_1 \times \cdots \times \Gamma_k, \delta, (q_{01}, \ldots, q_{0k}), (Z_{01}, \ldots, Z_{0k}), F_1 \times \cdots \times F_k \rangle$, where δ is a mapping from $(K_1 \times \cdots \times K_k) \times (\Sigma_1 \cup \{\epsilon\}) \times (\Gamma_1 \times \cdots \times \Gamma_k)$ into the finite subsets of $(K_1 \times \cdots \times K_k) \times (\Gamma_1^* \times \cdots \times \Gamma_k^*)$ defined as follows: For $(q_1, \ldots, q_k)$ in $K_1 \times \cdots \times K_k$, a_i in $(\Sigma_i \cup \{\epsilon\})$, $1 \leqslant i \leqslant k$, $(Z_1, \ldots, Z_k)$ in $\Gamma_1 \times \cdots \times \Gamma_k$, $\delta((q_1, \ldots, q_k), a_1, (Z_1, \ldots, Z_k))$ contains $((p_1, \ldots, p_k), y_1, \ldots, y_k)$ if (1) $\delta_i(q_i, a_i, Z_i)$ contains (p_i, y_i) for $1 \leqslant i \leqslant k$ and (2) $\phi_i(q_i, a_i, z_i) = a_{i+1}$ for $1 \leqslant$

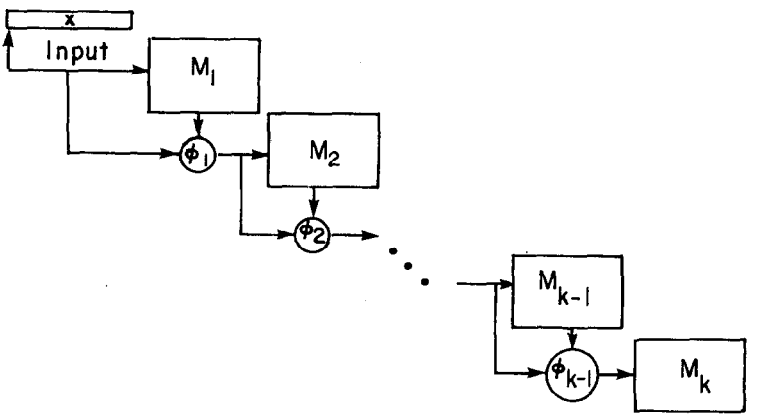

FIG. 3. $M_1\phi_1M_2\phi_2 \cdots \phi_{k-1}M_k$.

$i \leqslant k-1$. The language accepted by $M_1\phi_1M_2\phi_2 \cdots \phi_{k-1}M_k$ is the set of all strings x in Σ_1^* such that $M_1\phi_1M_2 \cdots \phi_{k-1}M_k$, when started in state $(q_{01}, \ldots, q_{0k})$ on the leftmost symbol of x (see Figure 3) with $Z_{01}, \ldots, Z_{0k}$ the topmost symbols of the pushdown stores, eventually reads all of x and enters a k-tuple of states $(q_1, \ldots, q_k)$ in $F_1 \times \cdots \times F_k$.

The following theorem is one of the main results in [15].

THEOREM 6.1. *Let M be a deterministic single-tape Turing machine. We can effectively construct a* PDA M_1, *a* DPDA M_2 *and a cascade mapping ϕ_1 such that $M_1\phi_1M_2$ accepts precisely the language accepted by M.*

The construction of $M_1\phi_1M_2$ in [15] is similar to the construction used to prove that a deterministic single-tape Turing machine M can be simulated by a deterministic one-way automaton with two pushdown stores [7]. Thus, $M_1\phi_1M_2$ operates in such a way that $M_1(M_2)$ simulates M when it is moving to the right (left). M_2 can see the topmost symbol of the pushdown store of M_1 via the mapping ϕ_1, but M_1 cannot see the topmost symbol of M_2. Hence, when M is moving right, M_1 is forced to "guess" the topmost symbol of M_2 and later check the guess via ϕ_1.

We shall see that Theorem 6.1 is not symmetric in that the language accepted by $M_1\phi_1M_2$ is recursive if M_1 is a DPDA and M_2 is a PDA. This answers an open problem in [15]. We shall actually prove a stronger result: The language accepted by $M_1\phi_1M_2\phi_2 \cdots \phi_{k-1}M_k$ is recursive provided $M_1, \ldots, M_{k-1}$ are DPDA's.

Notation. Let $M_1 = \langle K_1, \Sigma_1, \Gamma_1, \delta_1, q_{01}, Z_{01}, F_1\rangle$ be an ϵ-free DPDA (i.e., M_1 has no ϵ-moves.) Let ϕ_1: $K_1 \times \Sigma_1 \times \Gamma_1 \to \Sigma_2$ be a cascade mapping. We can regard each triple in $K_1 \times \Sigma_1 \times \Gamma_1$ as an abstract symbol and regard ϕ_1 as a homomorphism on $(K_1 \times \Sigma_1 \times \Gamma_1)^*$. For $L \subseteq \Sigma_1^*$, let $M_1(L) = \{\phi_1((q_1, a_1, Z_1) \cdots (q_n, a_n, Z_n)) \mid n \geqslant 1$, each a_i is in Σ_1, $a_1 \cdots a_n$ in L, there are states $q_1, \ldots, q_n$ with $q_1 = q_{01}$ and q_n in F_1, pushdown strings $\alpha_1Z_1, \ldots, \alpha_nZ_n$ (each α_i in Γ^*, Z_i in Γ) with $\alpha_1 = \epsilon$ and $Z_1 = Z_{01}$ such that for $1 \leqslant i < n$, M_1 in state q_i on input a_i and pushdown contents α_iZ_i enters state q_{i+1} with $\alpha_{i+1}Z_{i+1}$ the resulting pushdown contents$\} \cup \{\epsilon \mid$ if ϵ is in L and q_{01} is in $F_1\}$.

We will also need the following definition.

DEFINITION. A set $L \subseteq \Sigma_1^*$ is called *simple* if it is $\varnothing$ or it can be written as $L = L_1 \cup \cdots \cup L_r$, where each L_i is of the form $\{x\}$ or of the form $\{xy^j \mid j \geqslant 1\}$ where x is in Σ_1^* and y is in Σ_1^+. L is effectively computable if we can compute the specifications of all the L_i's (i.e., we can effectively find the x's and the y's of all L_i's).

The next proposition is obvious.

PROPOSITION 6.1. *Let L be an effectively computable simple set. Then we can construct a finite automaton accepting L.*

We now prove that if L is an effectively computable simple set, then so is $M_1(L)$.

LEMMA 6.1. *Let $M_1 = \langle K_1, \Sigma_1, \Gamma_1, \delta_1, q_{01}, Z_{01}, F_1\rangle$ be an ϵ-free* DPDA *and ϕ_1: $K_1 \times \Sigma_1 \times \Gamma_1 \to \Sigma_2$ be a cascade mapping. If $L \subseteq \Sigma_1^*$ is an effectively computable simple set, then so is $M_1(L)$.*

Proof. If $L = \varnothing$, then $M_1(L) = \varnothing$. So assume that $L = L_1 \cup \cdots \cup L_r$, $r \geqslant 1$, each L_i is of the form $\{x\}$ or of the form $\{xy^j \mid j \geqslant 1\}$ (x in Σ_1^*, y in Σ_1^+). Then $M_1(L) = M_1(L_1) \cup \cdots \cup M_1(L_r)$. Clearly, it is sufficient to show that each $M_1(L_i)$ is an effectively computable simple set. If L_i is of the form $\{x\}$, then $M_1(L_i)$ is either $\varnothing$ or a singleton and hence a simple set.

Now suppose that $L_i = \{xy^j \mid j \geqslant 1\}$. Assume that $x \neq \epsilon$. (The case when $x = \epsilon$ is similar.) Then $x = a_1 \cdots a_n$, $y = a_{n+1} \cdots a_m$ for some $n \geqslant 1$, $m \geqslant n + 1$. Construct an ϵ-free DPDA $\bar{M}_1 = \langle \bar{K}_1, \{0\}, \Gamma_1, \bar{\delta}_1, \bar{q}_{01}, Z_{01}, \bar{F}_1 \rangle$, where $\bar{K}_1 = \{[a_1 \cdots a_{j-1} q a_j \cdots a_m] \mid q \text{ in } K_1, 1 \leqslant j \leqslant m\}$, $\bar{q}_{01} = [q_0 a_1 \cdots a_m]$, $\bar{F}_1 = \{[a_1 \cdots a_{j-1} q a_j \cdots a_m] \mid q \text{ in } F_1, 1 \leqslant j \leqslant m\}$ and $\bar{\delta}_1$ is defined as follows: Suppose that $\delta_1(q, a_j, Z) = (p, y)$ then

$$\bar{\delta}_1([a_1 \cdots a_{j-1} q a_j \cdots a_m], 0, Z) = \begin{cases} ([a_1 \cdots a_j p a_{j+1} \cdots a_m], y) & \text{if } j < m \\ ([a_1 \cdots a_n p a_{n+1} \cdots a_m], y) & \text{if } j = m \end{cases}$$

Also define a cascade mapping $\bar{\phi}_1$: $\bar{K}_1 \times \{0\} \times \Gamma_1 \to \Sigma_2$ by $\bar{\phi}_1([a_1 \cdots a_{j-1} q a_j \cdots a_m], 0, Z) = \phi_1(q, a_j, Z)$.

Clearly, $M_1(L_i) = \bar{M}_1(0^*)$. Hence, it is sufficient to show that $\bar{M}_1(0^*)$ is an effectively computable simple set.

Since $\bar{M}_1$ is ϵ-free and has an alphabet consisting of only one symbol, 0, there are only two possibilities for the form of $\bar{M}_1(0^*)$:

Case 1. The set accepted by $\bar{M}_1$ is finite, in this case, $\bar{M}_1(0^*)$ is finite and hence a simple set.

Case 2. The set accepted by $\bar{M}_1$ is infinite. Then there must exist a unique least positive integer t such that 0^t is accepted by $\bar{M}_1$ and $t = r + s$ for some unique non-negative integers r and s with $s \geqslant 1$ satisfying: During the computation of $\bar{M}_1$ on 0^t exactly one of the following happens:

(a) The (state, pushdown store configuration) of $\bar{M}_1$ after reading 0^r is the same as that after reading 0^{r+s}.

(b) The (state, topmost symbol) of $\bar{M}_1$ after reading 0^r is the same as that after reading 0^{r+s}. Moreover, during this interval, the pushdown store does not decrease in length.

It follows that $\bar{M}_1(0^*) = L_1 \cup \{xy^j \mid j \geqslant 1\}$ where $L_1 \subseteq \{w \mid 0 \leqslant |w| \leqslant t - 1\}$[8] and $|x| = r$, $|y| = s$. Hence $\bar{M}_1(0^*)$ is a simple set.

To see that $\bar{M}_1(0^*)$ is effectively computable, we need only note that there is an upper-bound on t of case 2(b): $t \leqslant |\bar{K}_1| \cdot (|\Gamma_1|^{c|\Gamma_1|^{|\bar{K}_1|+1}})$, where $c = \max\{|y| \mid \bar{\delta}_1(q, 0, Z) = (p, y), (q, Z) \text{ in } \bar{K}_1 \times \Gamma_1\}$. ∎

We are now ready to prove the main result of this section.

THEOREM 6.2. *Let $k \geqslant 2$ and for $1 \leqslant i \leqslant k$, let $M_i = \langle K_i, \Sigma_i, \Gamma_i, \delta_i, q_{0i}, Z_{0i}, F_i \rangle$ be* DPDA*'s (M_k may be nondeterministic.) Let ϕ_i: $K_i \times (\Sigma_i \cup \{\epsilon\}) \times \Gamma_i \to \Sigma_{i+1} \cup \{\epsilon\}$ be*

[8] $|w|$ = length of w.

cascade mappings for $1 \leqslant i < k$. *Then the language accepted by* $M_1\phi_1M_2 \cdots \phi_{k-1}M_k$ *is recursive.*

Proof. First we construct from M_i ($1 \leqslant i \leqslant k$) a new machine M_i' by replacing the occurrences of 'ϵ' in the specification of δ_i by a new symbol '\$'. Thus, $M_i' = \langle K_i, \Sigma_i \cup \{\$\}, \Gamma_i, \delta_i', q_{0i}, Z_{0i}, F_i\rangle$, where δ_i' now has domain $K_i \times (\Sigma_i \cup \{\$\}) \times \Gamma_i$. (Hence, M_i' is ϵ-free.) Similarly, we define a cascade mapping ϕ_i' from ϕ_i by replacing 'ϵ' in the domain and range of ϕ_i by '\$', i.e., ϕ_i' is now a mapping from $K_i \times (\Sigma_i \cup \{\$\}) \times \Gamma_i \to \Sigma_{i+1} \cup \{\$\}$.

Let x be in Σ_1^*. We describe an algorithm to determine if x is accepted by $M_1\phi_1M_2\phi_2 \cdots \phi_{k-1}M_k$. We may assume that x is accepted by M_1, otherwise there is nothing to do. Let $L_x = \{y \mid y$ in $(\Sigma_1 \cup \{\$\})^*$, y is accepted by M_1' and y with \$'s deleted is equal to $x\}$. Clearly, x is accepted by $M_1\phi_1M_2\phi_2 \cdots \phi_{k-1}M_k$ if and only if $M_{k-1}'(\cdots M_1'(L_x)\cdots) \cap T(M_k') \neq \emptyset$. ($T(M_k')$ denotes the language accepted by M_k'.) Assume for the moment that L_x is an effectively computable simple set. Then by Lemma 6.1, $R_x = M_{k-1}'(\cdots M_1'(L_x)\cdots)$ is an effectively computable simple set and a finite automaton can be constructed accepting R_x. Then we can construct a PDA accepting the set $R_x \cap T(M_k')$. The result would then follow from the solvability of the emptiness problem for languages accepted from the solvability of the emptiness problem for languages accepted by PDA's [7].

We now show that L_x is an effectively computable simple set. Since M_1 and M_1' are deterministic there exists a w in $(\Sigma_1 \cup \{\$\})^*$ such that $L_x \subseteq \{w\$^i \mid i \geqslant 0\}$. We also note that $M_1(M_1')$ can enter an infinite loop on ϵ-input (\$-input), passing through accepting and nonaccepting states, only after reading all of $x(w)$. Since the computation of M_1' after reading w will only be on a string of \$'s, an argument similar to that of Lemma 6.1 would show that L_x is an effectively computable simple set. We leave the details to the reader. ∎

The following corollary solves an open problem in [15].

COROLLARY 6.1. *Let* M_1 *be a* DPDA, M_2 *a* PDA *and* ϕ_1 *a cascade mapping. Then the language accepted by* $M_1\phi_1M_2$ *is recursive.*

REFERENCES

1. R. W. EHRICH AND S. S. YAU, Two-way sequential transductions and stack automata, *Inform. Contr.* **18** (1971), 404–446.
2. S. GINSBURG, S. A. GREIBACH, AND M. A. HARRISON, One-way stack automata, *J. Assoc. Comput. Mach.* **14** (1967), 389–418.
3. S. GINSBURG AND E. H. SPANIER, Bounded Algol-like languages, *Trans. Amer. Math. Soc.* **113** (1964), 333–368.
4. S. A. GREIBACH, Checking automata and one-way stack languages, *J. Comput. System Sci.* **3** (1969), 196–217.
5. S. A. GREIBACH, One-way finite visit automata, *Theor. Comput. Sci.* **6** (1978), 175–221.
6. T. V. GRIFFITHS, The unsolvability of the equivalence problem for ϵ-free nondeterministic generalized machines, *J. Assoc. Comput. Mach.* **15** (1968), 409–413.

7. J. E. HOPCROFT AND J. D. ULLMAN, "Formal Languages and their Relation to Automata," Addison–Wesley, Reading, Mass., 1969.
8. O. H. IBARRA, On two-way sequential transductions of full semiAFL's, *Theor. Comput. Sci.* **7** (1978), 287–309.
9. O. H. IBARRA, Reversal-bounded multicounter machines and their decision problems, *J. Assoc. Comput. Mach.* **25** (1978), 116–133.
10. O. H. IBARRA, The unsolvability of the equivalence problem for ϵ-free NGSM's with unary input (output) alphabet and applications, *SIAM J. Comput.* **7** (1978), 524–532.
11. R. PARIKH, On context-free languages, *J. Assoc. Comput. Mach.* **13** (1966), 570–581.
12. M. RABIN AND D. SCOTT, Finite automata and their decision problems, *IBM J. Res. Develop.* **3** (1959), 114–125.
13. V. RAJLICH, Absolutely parallel grammars and two-way finite-state transducers, *J. Comput. System Sci.* **6** (1972), 324–342.
14. L. VALIANT AND M. PATTERSON, Deterministic one-counter automata, *J. Comput. System Sci.* **10** (1975), 340–350.
15. T. AE, Direct or cascade product of pushdown automata, *J. Comput. System Sci.* **14** (1977), 257–263.