Theoretical Computer Science 24 (1983) 313–322
North-Holland

ON SOME DECISION QUESTIONS CONCERNING PUSHDOWN MACHINES*

Oscar H. IBARRA

Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Communicated by Michael A. Harrison
Received June 1981
Revised March 1982

Abstract. Certain decision questions (e.g., equivalence, ambiguity, single-valuedness, monotonicity, etc.) concerning pushdown transducers and acceptors are investigated. Examples of results shown are the following:

(1) There is an algorithm to decide given two deterministic pushdown transducers, one of which has a 'finite-turn' counter for a pushdown stack, whether they are equivalent (i.e., they define the same input/output relation).

(2) There are subclasses of pushdown acceptors which have a decidable ambiguity problem but an undecidable inherent ambiguity problem. (It seems that these classes are the first known to exhibit this property.)

(3) There is an algorithm to decide given a homomorphism h and a language L accepted by a finite-turn nondeterministic counter automaton whether h is one-to-one on L. In contrast, there is no such algorithm if L is accepted by a 1-turn deterministic pushdown automaton or by an unrestricted deterministic counter automaton.

1. Introduction

An important unresolved question in formal language theory is whether equivalence is decidable for the class of deterministic pushdown automata (DPDA's). Equivalence is decidable for several restricted classes (see, e.g., [6, 15, 17–19, 22–24]). Among these classes are the following:

(1) *Finite-turn DPDA's* [23]. (A DPDA is finite-turn if the number of alternations between pushing and popping the acceptor makes on any input is bounded by a given positive integer.)

(2) *Deterministic counter automata* (DCA's) [24]. (A DCA is a PDA which has exactly 2 pushdown symbols one of which is used only as a bottom-of-stack marker.)

It has also been recently shown that equivalence of a DCA (or finite-turn DPDA) M_1 and a DPDA M_2 is decidable [20].

In this paper, we show that there is an algorithm to decide given a deterministic pushdown tranducer (DPDT) M_1 and a finite-turn deterministic counter transducer (finite-turn DCT) M_2 whether they are equivalent (i.e., they define the same

* This research was supported in part by NSF Grant MCS8102853.

input/output relation). We also consider related problems concerning ambiguity, single-valuedness, one-to-oneness and monotonicity. For example, we show the following:

(i) It is decidable to determine if a finite-turn nondeterministic counter automaton (finite-turn NCA) is ambiguous.

(ii) It is undedicable to determine if the language accepted by an ε-free 1-turn NCA[1] is inherently ambiguous (i.e., not recognizable by any unambiguous ε-free 1-turn NCA).

Thus the class of ε-free 1-turn NCA's has a decidable ambiguity problem but an undecidable inherent ambiguity problem. (Hence, the two problems are independent.) We know of no other class having this property. The problems in (i) and (ii) are undecidable for ε-free NCA's and ε-free 1-turn NPDA's. Some of our results generalize or improve previously known results.

2. Definitions

A *nondeterministic pushdown transducer* (NPDT) [1] is an 8-tuple

$$M = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F),$$

where Q, Σ, Γ and F are the sets of states, input alphabet, pushdown alphabet and accepting states, respectively; q_0 is the start state and Z_0 in Γ is the bottom-of-stack marker which appears only at the bottom of the stack and is never changed; Δ is the output alphabet and δ is a mapping from $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^* \times \Delta^*$ (ε denotes the null string).

The translation defined by M, denoted by $R(M)$, is the set of all pairs (x, y) such that M when started at the left end of x in state q_0 with Z_0 on the pushdown stack, scans *all* symbols of x and enters an accepting state after outputing y. M is *deterministic* (DPDT) if for all q in Q and Z in Γ, $\delta(q, \varepsilon, Z) \neq \emptyset$ implies $\delta(q, a, Z) = \emptyset$ for all a in Σ. Note that for a given x there may be more than one (possibly infinitely many) y such that (x, y) is in $R(M)$, even when M is deterministic. (This is because we allow ε-moves, i.e., moves on ε input.) The language accepted by M, denoted by $L(M)$, is the set of all x such that (x, y) is in $R(M)$. An NCT (DCT) is an NPDT (DPDT) which has exactly 2 pushdown symbols, Z_0 and Z_1, but Z_0 only appears at the bottom of the stack and is never changed. (Thus the pushdown can only be used as a *counter*.) When the counter is deleted from an NCT it becomes a *nondeterministic finite transducer* [1] (NFT). Similarly, a *deterministic finite transducer* (DFT) is a DCT without the counter. When the outputs are deleted, an NPDT (DPDT, NCT, DCT, NFT, DFT) becomes an NPDA (DPDA, NCA, DCA, NFA, DFA). An NPDT is finite-turn if the number of alterations between pushing and popping on any input is bounded by some given number. When the bound is 1, it is 1-turn. Finite-turn NCT, NPDA, etc., are defined similarly.

[1] ε-free means no ε-moves.

Remark. It would be easier to remember what the various abbreviations for the machines denote if we note that N stands for 'nondeterministic', D for 'deterministic', PD for 'pushdown', C for 'counter', F for 'finite', T for 'transducer' and A for 'automaton' or 'acceptor'.

3. Equivalence

The following theorem strengthens a result in [8]. In [8] the theorem is shown only for the case when the deterministic transducers have *halting* accepting states (thus, for each x, there is at most one y such that (x, y) is in $R(M)$). Our proof uses ideas of [8, 14].

Theorem 3.1. *There is an algorithm to decide given a* DPDT M_1 *and a finite-turn* DCT M_2 *whether* $R(M_1) = R(M_2)$.

Proof. The proof uses the following result in [14]: There is an algorithm to decide given an NPDA M augmented with finite-turn counters whether $L(M)$ is empty.

Now given a DPDT M_1 and a finite-turn DCT M_2, define the language L consisting of all strings $\#x\#$ (# is a new symbol) satisfying one of the following conditions:

(1) M_i accepts x but M_{3-i} does not $(i = 1, 2)$.

(2) Both M_1 and M_2 accept x, but for some (x, y_1) in $R(M_1)$ and (x, y_2) in $R(M_2)$ and $k \geq 1$, the kth symbol of $y_1 \neq$ the kth symbol of y_2.

Clearly, $R(M_1) = R(M_2)$ if and only if L is empty.

We can construct an NPDA M augmented with three finite-turn counters such that $L(M) = L$. M when given input $\#x\#$ initially stores in its first two counters a positive integer k, chosen nondeterministically. Then M simulates M_1 and M_2 in parallel on input x. The pushdown stack of M simulates the pushdown of M_1 while the third counter simulates the counter of M_2. M accepts $\#x\#$ if and only if (1) or (2) above is true. At each step of the simulation of M_i, counter i is decremented by an amount equal to the length of the output string corresponding to the move. When counter i becomes zero, the kth symbol of the output M_i has generated so far is recorded in the finite control. Thus, the 'discrepancy' in the kth symbols described in (2) above can be verified. Because of ε-moves, the parallel simulation of M_1 and M_2 by M is not straightforward. However, using techniques similar to the ones used to show that languages accepted by DPDA's are closed under complementation [13], the simulation can be carried out correctly. □

Remark. If, in Theorem 3.1, M_1 or M_2 is nondeterministic, then the equivalence problem is undecidable. In fact, it is undecidable to determine given an ε-free 1-turn NCA M whether $L(M) = \Sigma^*$ [2, 7].

Notation. Let $n \geq 1$. An n-NPDT is an NPDT augmented with n finite-turn

counters. $n-$DPDT, $n-$NPDA, $n-$DPDA, $n-$NFT, $n-$DFT, $n-$NFA and $n-$DFA are defined similarly. Note that a $1-$NFT ($1-$DFT, etc.) is the same as a finite-turn NCT (finite-turn DCT, etc.).

Theorem 3.1 is easily generalized as follows.

Corollary 3.2. *There is an algorithm to decide given an* $n-$DPDT M_1 *and an* $m-$DFT M_2 *whether* $R(M_1)=R(M_2)$.

Corollary 3.3. *There is an algorithm to decide given an* $n-$DPDA M_1 *and an* $m-$DFA M_2 *whether* $L(M_1)=L(M_2)$.

4. Ambiguity and single-valuedness

We will be using the following known facts.

Fact 1. We can effectively construct for a given Turing machine (TM) M two ϵ-free[2] DCA's M_1 and M_2 and a homomorphism h such that $L(M)=h(L(M_1)\cap L(M_2))$ (see [10, Theorem 1 and its corollary on p. 373]).

Fact 2. This is the same as Fact 1, but now M_1 and M_2 are ϵ-free 1-turn DPDA's [2].

Fact 3. There is no algorithm to decide given two ϵ-free DCA's (or two ϵ-free 1-turn DPDA's) M_1 and M_2 whether $L(M_1)\cap L(M_2)=\emptyset$. (This follows from Facts 1 and 2 and the undecidability of the halting problem for TM's.)

An acceptor (or transducer) M is *ambiguous* if there is an input string in $L(M)$ which has at least 2 distinct accepting computations. A transducer M is *single-valued* if for each input x there is at most one y such that (x, y) is in $R(M)$. Note that an unambiguous transducer is single-valued but the converse is not true in general.

Proposition 4.1. *The ambiguity problem (deciding if a machine is ambiguous) is undecidable for* ϵ-free NCA's *and* ϵ-*free* 1-*turn* NPDA's.

Proof. This follows from Fact 3 and the observation that we can construct from M_1 and M_2 an ϵ-free NCA (ϵ-free 1-turn NPDA) M which nondeterministically chooses one of the M_i's to simulate. Then M is ambiguous if and only if $L(M_1)\cap L(M_2)\neq\emptyset$. □

[2] Thus, each computation (accepting or not) is unique.

Proposition 4.2. *The single-valuedness problem (deciding if a transducer is single-valued) is undecidable for ε-free* NCT*'s and ε-free* 1*-turn* NPDT*'s.*

Proof. The proof is the same as in Proposition 4.1, but now M also outputs, at each step, the transition rule used in the move. (The transition rules can be coded in binary.) Then M is single-valued if and only if $L(M_1) \cap L(M_2) = \emptyset$. □

Note that the proof of Proposition 4.2 also shows that the ambiguity problem is reducible to the single-valuedness problem. In contrast to Proposition 4.2, we have the following.

Theorem 4.3. *There is an algorithm to decide given an* $n-$NFT M *whether* M *is single-valued.*

Proof. The proof is similar to that of Theorem 3.1. For convenience, let M_1 and M_2 be 2 copies of M. Define the language L consisting of all strings $\#x\#$ such that both M_1 and M_2 accept x, but for some (x, y_1) in $R(M_1)$ and (x, y_2) in $R(M_2)$ and $k \geq 1$, the kth symbol of $y_1 \neq$ the kth symbol of y_2. We can construct a $(2n+2)-$NFA M' accepting L, and $L = L(M')$ is empty if and only if M is single-valued. □

Corollary 4.4. *There is an algorithm to decide given an* $n-$NFT M *and a context-free language* (CFL) L *whether* M *is single-valued on* L *(i.e., for each* x *in* L*, there is at most one* y *such that* (x, y) *is in* $R(M)$*).*

Proof. The acceptor M' in the proof of Theorem 4.3 is augmented with a pushdown stack which enables M' to check that x is in L. The result follows since the emptiness problem for such machines is decidable (see Theorem 3.1). □

Corollary 4.5. *There is an algorithm to decide given an* $n-$NFT M *and a* CFL L *whether* M *is ambiguous on* L*. Thus, the ambiguity problem for* $n-$NFT*'s and* $n-$NFA*'s is decidable (this corresponds to the case when* $L = \Sigma^*$*).*

Remark. Corollaries 4.4 and 4.5 are still valid for the case when L is a language accepted by an $m-$NPDA.

Although the single-valuedness problem for $n-$NFT's is decidable, the equivalence problem is undecidable. In fact, we have the following.

Proposition 4.6. *It is undecidable to determine given an ε-free* 1*-turn* NCT M *with input and output alphabets* Σ *and* $\{1\}$*, respectively, whether* $R(M) = \Sigma^+ \times \{1\}$*. (Note that* $\Sigma^+ \times \{1\}$ *can be computed by a deterministic* 2*-state* gsm.)

Proof. This follows from the fact that there is no algorithm to decide given an ε-free 1-turn NCA M' over input alphabet Σ whether $L(M') = \Sigma^+$ [2, 7]. □

However, we can prove the following result which is stronger than [5, Proposition 5].

Theorem 4.7. *There is an algorithm to decide given an* $n-$NPDT M_1 *and an* $m-$NFT M_2, *satisfying* (a) M_1 *or* M_2 *is single-valued, and* (b) $L(M_1) = L(M_2)$, *whether* $R(M_1) = R(M_2)$.

Proof. The construction of M from M_1 and M_2 in the proof of Theorem 3.1 applies. However, M need only check condition (2). □

The next corollary is stronger than [5, Proposition 5].

Corollary 4.8. *There is an algorithm to decide given an* $n-$NPDT M_1 *and an* NFT M_2, *satisfying* (a) M_1 *or* M_2 *is single-valued, and* (b) $L(M_2) \subseteq L(M_1)$, *whether* $R(M_1) = R(M_2)$.

Proof. This follows from Theorem 4.7 and the observation that $L(M_1) \stackrel{?}{\subseteq} L(M_2)$ is decidable since $L(M_2)$ is a regular set. □

Our next corollary is equivalent to [5, Proposition 4] (see also [3]).

Corollary 4.9. *There is an algorithm to decide given an unambiguous* NPDT M_1 *and an* NFT M_2 *whether* $R(M_1) = R(M_2)$.

Proof. The proof follows from Theorem 4.7 and the fact that $L(M_1) = L(M_2)$ is decidable for unambiguous M_1 [21]. □

Let C be a class of acceptors. A langauge L accepted by a machine in C is *inherently ambiguous* (with respect to C) if every machine in C accepting L is ambiguous. The *inherent ambiguity problem* for C is the problem of deciding given an acceptor M in C whether $L(M)$ is inherently ambiguous. (Recall that the *ambiguity problem* is the problem of deciding given M in C whether M is ambiguous.)

The next theorem and the remark following it show that there are subclasses of pushdown acceptors which have a decidable ambiguity problem but an undecidable inherent ambiguity problem. As far as we know these classes are the first to exhibit such a property.

Theorem 4.10. *Let* C_1 *be the class of* ε-*free* 1-*turn* NCA's. *Then*

(1) *the ambiguity problem for* C_1 *is decidable,*

(2) *the inherent ambiguity problem for* C_1 *is undecidable.*

Proof. (1) follows directly from Corollary 4.5. To prove (2), let $\mathscr{L}_1$ denote the class of languages accepted by machines in C_1. The following statements are known or easily verified:

(1) $L_0 = \{a^i b^j c^k \mid i, j, k \geq 1, i = j \text{ or } j = k\}$ is in $\mathscr{L}_1$, and L_0 is inherently ambiguous. (In fact, L_0 is inherently ambiguous with respect to NPDA's [9].)

(2) The universe problem (i.e., deciding for L in $\mathscr{L}_1$ whether $L = \Sigma^*$) is undecidable [2, 7].

(3) $\mathscr{L}_1$ is effectively closed under concatenation with regular sets and union.

(4) For each L in $\mathscr{L}_1$ and symbol a, L is inherently ambiguous if and only if $L/a = \{x \mid xa \text{ in } L\}$ is inherently ambiguous. (Thus, inherent ambiguity is preserved.)

Statements (1)–(4) above satisfy the hypothesis of Greibach's theorem [13]. Hence, the inherent ambiguity problem for C_1 is undecidable. □

Theorem 4.10 holds for other classes of acceptors. For example, it holds for (ε-free) n-NFA's.

5. One-to-oneness and monotonicity

An NCT (NPDT) M is *one-to-one* if (x_1, y) and (x_2, y) in $R(M)$ implies $x_1 = x_2$. The next proposition implies that the one-to-oneness problem is undecidable.

Proposition 5.1. *It is undecidable to determine given an ε-free* DCA *(or an ε-free* 1*-turn* DPDA*) M and a length-preserving homomorphism*[3] *h whether h is one-to-one on $L(M)$.*

Proof. Let M_1 and M_2 be ε-free DCA's. For $i = 1, 2$, let Σ_i be a set of abstract symbols representing the transition rules of M_i. Let $\Sigma = \Sigma_1 \cup \Sigma_2$. We construct an ε-free DCA M with input alphabet Σ. M when given input x in Σ^* checks whether x is in Σ_1^+ or in Σ_2^+. While checking, M simulates the computation of the appropriate M_i (as dictated by the string x). M enters an accepting state if and only if M_i enters an accepting state during the simulation. Now define the length-preserving homomorphism h by:

$$\text{For each } \alpha \text{ in } \Sigma, \quad h(\alpha) = a,$$

where a is the input symbol associated with the transition rule α. Then h is one-to-one on $L(M)$ if and only if $L(M_1) \cap L(M_2) = \emptyset$. The result follows by Fact 3. □

Remark. It was pointed out in [4] (see also [11]) that the problem of deciding if an NFT is one-to-one on a given CFL is undecidable. Proposition 5.1 is a stronger result.

[3] A homomorphism h on Σ^* is length-preserving if $h(a)$ is a single symbol for each symbol a in Σ.

Corollary 5.2. *It is undecidable to determine given an ε-free* DCT *(or an ε-free 1-turn* DPDT*) M whether M is one-to-one.*

For $n-$NFT's we have the following result which is stronger than [3, Corollary 2]. (In [3] the result is shown only for NFT's.)

Theorem 5.3. *There is an algorithm to decide given an $n-$*NFT *M whether M is one-to-one.*

Proof. Let M_1 and M_2 be 2 copies of M. We construct a $(2n+2)-$NFA (i.e., a nondeterministic finite automaton with $2n+2$ finite-turn counters) M' which operates as follows: Given input y, M' nondeterministically guesses 2 strings x_1 and x_2 (symbol by symbol) and simulates (in parallel) the computations of M_1 and M_2 on x_1 and x_2, respectively. M' accepts y if and only if (x_1, y) is in $R(M_1)$, (x_2, y) is in $R(M_2)$ and $x_1 \neq x_2$. Now M' has $2n+2$ finite-turn counters; $2n$ of these counters are used in the simulation of M_1 and M_2. The two extra counters are used to check that the 'guessed' strings x_1 and x_2 are different. Note that $x_1 \neq x_2$ if and only if $|x_1| \neq |x_2|$ or[4] for some k, the kth symbol of $x_1 \neq$ the kth symbol of x_2. Thus, the two counters can be used to verify that $x_1 \neq x_2$. The result follows since we can decide whether $L(M') = \emptyset$. □

Corollary 5.4. *There is an algorithm to decide given an $n-$*NFT *M_1 and an $m-$*NFA *M_2 whether M_1 is one-to-one on $L(M_2)$.*

Finally, we consider the question of monotonicity. A transducer M is *monotonic* if for each pair (x, y) in $R(M)$, $|y| \geq |x|$. Monotonicity of transducers is related to some problems concerning Lindenmayer schemes (see, e.g., [12]). There is an algorithm to decide if an NFT is monotonic on a CFL [16]. The next theorem strengthens this result.

Theorem 5.5. *There is an algorithm to decide each of the following problems:*

(1) *Given an $n-$*NPDT *M, is M monotonic?*

(2) *Given an $n-$*NPDT *M_1 and an $m-$*NFA *M_2, is M_1 monotonic on $L(M_2)$?*

(3) *Given an $n-$*NFT *M_1 and an $m-$*NPDA *M_2, is M_1 monotonic on $L(M_2)$?*

Proof. We just prove (1). The proofs of (2) and (3) are similar. Given an $n-$NPDT M, we construct an $(n+2)-$NPDA M' which on input x simulates the computation of M on x. M' accepts x if and only if (x, y) is in $R(M)$ for some $|y| < |x|$. M' can check if $|y| < |x|$ by using two 1-turn counters. □

The next proposition shows that Theorem 5.5 cannot be improved.

[4] $|x|$ denotes the length of x.

Proposition 5.6. *There is no algorithm to decide given an ε-free transducer M_1 and an ε-free acceptor M_2 whether M_1 is monotonic on $L(M_2)$ for*

(1) *M_1 a 1-turn* DPDT *and M_2 a 1-turn* DPDA.
(2) *M_1 a 1-turn* DPDT *and M_2 a* DCA.
(3) *M_1 a* DCT *and M_2 a* DCA.
(4) *M_1 a* DCT *and M_2 a 1-turn* DPDA.

Proof. This follows from Fact 3 stated at the beginning of Section 4 and Proposition 5.7 below. □

Proposition 5.7. *We can effectively construct for a given* TM *M an ε-free* DCA *M_1, an ε-free 1-turn* DPDA *M_2, and a homomorphism h such that $L(M) = h(L(M_1) \cap L(M_2))$. Hence, there is no algorithm to decide given an ε-free* DCA *M_1 and an ε-free 1-turn* DPDA *M_2 whether $L(M_1) \cap L(M_2) = \emptyset$.*

Proof. Let M be a TM. Then we can effectively construct ε-free DCA's M_1 and M_2 and a homomorphism h such that $L(M) = h(L(M_1) \cap L(M_2))$ [10] (see Fact 1). Let Σ be the input alphabet of M_1 and M_2. Let $\bar{\Sigma} = \{\bar{a} \mid a \text{ in } \Sigma\}$ and g be a homomorphism defined by $g(a) = \bar{a}$ for each a in Σ. Let

$$A = \{x \# g(y^R) \mid x \text{ in } L(M_1) \text{ and } y \text{ in } L(M_2)\},$$

where $\#$ is a new symbol and y^R is the reverse of y. Then A can be accepted by an NCA M_3. Let $B = \{w \# g(w^R) \mid w \text{ in } \Sigma^*\}$. Clearly B can be accepted by a 1-turn DPDA M_4. Now extend the domain of h by defining $h(\#) = \varepsilon$ and $h(\bar{a}) = \varepsilon$ for each $\bar{a}$ in $\bar{\Sigma}$. Then $L(M) = h(L(M_3) \cap L(M_4))$. By adding new symbols to the input alphabet of M_3 and M_4 for 'padding' and 'dictating' the moves of the machines, we can construct from M_3, M_4 and h an ε-free DCA M_5, an ε-free 1-turn DPDA M_6 and a homomorphism f such that $L(M) = f(L(M_5) \cap L(M_6))$. □

The proof above can be modified and the symbols coded in binary to get the following corollary.

Corollary 5.8. *Let $P = \{x \# x^R \mid x \text{ in } \{0, 1\}^*\}$. There is no algorithm to decide given an ε-free* DCA *M_1 whether $L(M_1) \cap P = \emptyset$.*

6. Remarks

(a) Corollary 5.8 also holds when M_1 is an ε-free 1-turn DPDA. The proof is a simple modification of [2, Theorem 1].

(b) There is no algorithm to decide given an ε-free DCA (or an ε-free 1-turn DPDA) M whether $P \subseteq L(M)$. This follows from Corollary 5.8 and remark (a) and closure under complementation.

(c) In contrast to remark (b) it is known that $L(M) \stackrel{?}{\subseteq} P$ is decidable for any NPDA M (see H.B. Hunt III and J.L. Rangel, Decidability of equivalence, containment, intersection, and separability of context-free languages, *Proc. 16th Ann. Symp. on Found. of Comput. Sci.*, 144–150). This result also follows from the observation that $L(M) \subseteq P$ if and only if $L(M) \cap \bar{P} = \emptyset$, and $\bar{P}$ can be accepted by a 1-turn NCA. Hence, $L(M) \cap \bar{P}$ can be accepted by an NPDA augmented with a 1-turn counter, and therefore emptiness of $L(M) \cap \bar{P}$ is decidable (see the proof of Theorem 3.1). Generalizing,

$$L(M_1) \stackrel{?}{\subseteq} \overline{L(M_2)} \text{ is decidable for any } n-\text{NPA } M_1 \text{ and } m-\text{NFA } M_2.$$

References

[1] A.V. Aho and J.D. Ullman, *The Theory of Parsing, Translation and Compiling, Vol. 1: Parsing* (Prentice-Hall, Englewood Cliffs, NJ, 1972).

[2] B. Baker and R.V. Book, Reversal-bounded multipushdown machines, *J. CSS* **8** (1974) 315–332.

[3] K. Culik II, Some decidability results about regular and pushdown translations, *Inform. Process. Lett.* **8** (1979) 5–8.

[4] K. Culik II, Homomorphisms: decidability, equality and test sets, in: *Formal Language Theory, Perspectives and Open Problems* (Academic Press, New York, 1980).

[5] K. Culik II and C. Choffrut, Classes of tranducers and their properties, Tech. Rept., Univ. of Waterloo, 1981.

[6] E.P. Friedman and S.A. Greibach, Superdeterministic DPDA's: The method of accepting does affect decision problems, *J. CSS* **19** (1979) 79–117.

[7] S.A. Greibach, An infinite hierarchy of context-free languages, *J. ACM* **16** (1969) 91–106.

[8] E.M. Gurari, Transducers with decidable equivalence problem, submitted for publication.

[9] M.A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, Reading, MA, 1978).

[10] J. Hartmanis and J.E. Hopcroft, What makes some language theory problems undecidable, *J.CSS* **4** (1970) 368–376.

[11] T. Head, Unique decipherability relative to a language, unpublished manuscript, 1979.

[12] T. Head, A-transducers and the monotonicity of IL schemes, *J. CSS* **21** (1980) 87–91.

[13] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Reading, MA, 1979).

[14] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, *J. ACM* **25** (1978) 116–133.

[15] A.J. Korenjak and J.E. Hopcroft, Simple deterministic languages, *IEEE 7th Symp. on Switching and Automata Theory*, Berkeley (1966) pp. 36–46.

[16] J. Leguy-Cordellier, Transductions rationnelles décroissantes et substitution, Thèse 3ème cycle, Université de Lille, 1980.

[17] M. Linna, Two decidability results for deterministic pushdown automata, *J. CSS* **18** (1979) 92–107.

[18] M. Oyamaguchi and N. Honda, The decidability of equivalence for deterministic stateless pushdown automata, *Inform. Contr.* **38** (1978) 367–376.

[19] M. Oyamaguchi, Y. Inagaki and N. Honda, The equivalence problem for real-time strict deterministic languages, *Inform. Contr.* **45** (1980) 90–115.

[20] M. Oyamaguchi, Y. Inagaki and N. Honda, The equivalence problem for 2dpda's, one of which is a finite-turn or one-counter machine, *J. CSS* **23** (1981) 366–382.

[21] A. Salomaa and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series* (Springer, Berlin, 1978).

[22] L.G. Valiant, Decision procedures for families of deterministic pushdown automata, Ph.D. Thesis, University of Warwick, 1973.

[23] L.G. Valiant, The equivalence problem for deterministic finite-turn pushdown automata, *Inform. Contr.* **25** (1974) 123–133.

[24] L.G. Valiant and M.S. Paterson, Deterministic one counter automata, *J. CSS* **10** (1975) 340–350.