Journal of Computer and System Sciences **65,** 278–294 (2002)
doi:10.1006/jcss.2002.1836

Some Decision Problems Concerning Semilinearity and Commutation[1]

Tero Harju[2]

Department of Mathematics, and Turku Centre for Computer Science, University of Turku, FIN-20014 Turku, Finland
E-mail: harju@utu.fi

Oscar Ibarra[3]

Department of Computer Science, University of California, Santa Barbara, California 93106
E-mail: ibarra@cs.ucsb.edu

Juhani Karhumäki

Department of Mathematics, and Turku Centre for Computer Science, University of Turku, FIN-20014 Turku, Finland
E-mail: karhumak@cs.utu.fi

and

Arto Salomaa

Turku Centre for Computer Science, Lemminkäisenkatu 14A, FIN-20520 Turku, Finland
E-mail: asalomaa@cs.utu.fi

Received March 30, 2001; revised February 01, 2002

Let $\mathscr{M}$ be a class of automata (in a precise sense to be defined) and $\mathscr{M}_c$ the class obtained by augmenting each automaton in $\mathscr{M}$ with finitely many reversal-bounded counters. We show that if the languages defined by $\mathscr{M}$ are effectively semilinear, then so are the languages defined by $\mathscr{M}_c$, and, hence, their emptiness problem is decidable. We give examples of how this result can be used to show the decidability of certain problems concerning the equivalence of morphisms on languages. We also prove a surprising undecidability result for commutation of languages: given a fixed two-element code K, it is undecidable whether a given context-free language L commutes with K, i.e., $LK = KL$. © 2002 Elsevier Science (USA)

Key Words: reversal-bounded counters; context-free languages; combinatorics on words; commutation of languages; morphisms.

[1]Supported under the Grant 44087 of the Academy of Finland.
[2]To whom correspondence should be addressed.
[3]Supported in part by NSF Grants IRI-9700370 and IIS-0101134.

0022-0000/02 $35.00

1. INTRODUCTION

We shall consider various combinatorial decision problems for families of languages accepted by classes of automata augmented with reversal-bounded counters. In particular, we generalize the decidability results of Parikh [18] and Ibarra [13] that have turned out to be very useful tools in proving other decidability results concerning language families. The classical result in [18] states that the commutative image of a context-free language is semilinear, and that it can be effectively constructed from the pushdown automaton (or the context-free grammar) defining the language. According to a result of [13], one can decide the emptiness problem for the languages accepted by one-way nondeterministic pushdown automata (respectively, finite automata) augmented with reversal-bounded counters. The proof consists of showing that such automata accept only semilinear sets, and the semilinear set can be effectively constructed from the automaton.

In Section 3, we prove a general result concerning semilinearity of languages accepted by automata augmented with reversal-bounded counters. Essentially, it is shown there that if the languages defined by a class $\mathscr{M}$ of automata are effectively semilinear, then so are the languages defined by the automata that are obtained from those of $\mathscr{M}$ by augmenting reversal-bounded counters. Several examples of $\mathscr{M}$ are given in Section 4.

The established decidability result for semilinearity is exploited in Section 5 to show the decidability of the *multiple equivalence problem* of morphisms for a large family of languages. In this problem, one asks for a finite set of morphisms h_1, $h_2, \ldots, h_k$ and a language L whether each word $w \in L$ is identified by a pair of these morphisms, $h_i(w) = h_j(w)$ with $i \neq j$.

In Section 6, we apply the result to deterministic context-free languages L and regular codes R to show that it is decidable whether L can be expressed as $L = \bigcup_{i \in I} R^i$ for some set $I \subseteq N$. In contrast to this result, we show that the above question is undecidable for context-free languages and two-element codes. From this later result it follows, by a result of [4], that if K is a given two-element code, then it is undecidable whether a given context-free language L commutes with K, that is, whether the language equation $LK = KL$ holds true.

We also present several open problems that involve the mentioned questions on semilinearity, language equivalence and commutation of languages.

2. PRELIMINARIES

We refer to [19] or [11] for the basic definitions on automata and languages, and to [2] or [17] to those on words.

Let Σ be an alphabet, that is, a finite set of symbols, and denote by Σ^* the set of all words over Σ including the empty word, denoted by ε. Let $w \in \Sigma^*$ be a word. Then $|w|$ denotes the length of w, and $|w|_a$ denotes the number of occurrences of the letter $a \in \Sigma$ in w.

For a language $L \subseteq \Sigma^*$, we denote by $\bar{L}$ its complement, $\Sigma^* \setminus L$, and let $L^+ = \{w_1 w_2 \ldots w_i \mid w_i \in L, i \geqslant 1\}$ be its Kleene closure. Denote $L^* = L^+ \cup \{\varepsilon\}$. A language

L is a *code*, if each word $w \in L^+$ has a unique factorization $w = w_1 w_2 \dots w_n$ in terms of elements $w_i \in L$.

The *shuffle* $u \sqcup v$ of two words $u, v \in \Sigma^*$ is a finite set consisting of the words $u_1 v_1 \dots u_k v_k$, where $u = u_1 u_2 \dots u_k$ and $v = v_1 v_2 \dots v_k$ for some $u_i, v_i \in \Sigma^*$. If L and K are two languages, their *shuffle* is the language

$$L \sqcup K = \bigcup_{u \in L, v \in K} u \sqcup v.$$

In this paper, *a language family* will always mean a class $\mathscr{L}$ of languages that are accepted (or defined) by the automata from a class of automata $\mathscr{M}$. That is, if $\mathscr{L}$ is a language family, then each of its element $L \in \mathscr{L}$ is specified by $L = L(M)$ for some $M \in \mathscr{M}$.

Let P be a (binary) language operation, i.e., P maps two languages L_1 and L_2 to a language $P(L_1, L_2)$. The family $\mathscr{L}$ is *effectively closed* under the operation P, if for any $L_i = L(M_i)$ with $M_i \in \mathscr{M}$ for $i = 1, 2$, $P(L_1, L_2) \in \mathscr{L}$ and one can effectively construct an $M \in \mathscr{M}$ such that $L(M) = P(L_1, L_2)$.

Our main concern are the language families that are accepted by multitape Turing machines, where the worktapes have restricted behaviour. The storage types that come into use are variants of the pushdown storages (first in, last out), counters (pushdown storages with one letter), and queues (first in, first out), the behaviour of which may be further restricted.

We adopt the following notation: if $\mathscr{M}$ is a class of automata, then

$$\mathscr{L}(\mathscr{M}) = \{L(M) \mid M \in \mathscr{M}\}$$

denotes the family of languages accepted by the automata in $\mathscr{M}$. For instance, $\mathscr{L}(PDA)$ denotes the family of context-free languages, that is, the languages accepted by one-way pushdown automata (PDAs).

Consider a class $\mathscr{M}$ of automata, where each $M \in \mathscr{M}$ is a nondeterministic finite automaton possibly augmented with a data structure (consisting of possibly Turing tapes with restricted behaviour). Formally, $M = \langle Q, \Sigma, q_0, F, D, \delta \rangle$, where Q is the finite state set, Σ is the input alphabet, q_0 is the start state, F is the set of accepting states, D is the data structure (consisting of k worktapes with alphabets Γ_i for $i = 1, 2, \dots, k$), and δ is the (multiple valued) transition function. The automaton M, which is nondeterministic, has finitely many transitions (i.e., moves) of the form

$$(p, act) \in \delta(q, a, loc),$$

where

- $q \in Q$ is the current state, $a \in \Sigma \cup \{\varepsilon\}$ is the input read, and $loc = (a_1, \dots, a_k)$ is the "local" portion of the data structure D that influences (affects) the move, where $a_i = \varepsilon$ or $a_i \in \Gamma_i$ is the letter that is read on the ith worktape;
- $p \in Q$ is the state entered by M, and $act = (\alpha_1, \alpha_2, \dots, \alpha_k)$ is the action that M performs on the worktapes.

For example, if D consists of a pushdown stack and a queue, then $loc = (a, b)$, where a is the top symbol of the stack and b the symbol in the front of the queue or ε

if the queue is empty; and $act = (\alpha_1, \alpha_2)$, where α_1 pops the top symbol and pushes a word (possibly empty) onto the stack, and α_2 deletes the front of the queue (if $b \neq \varepsilon$) and possibly adds a symbol to the rear of the queue.

3. SEMILINEARITY AND AUGMENTED COUNTERS

Let $\boldsymbol{N}$ be the set of nonnegative integers and k be a positive integer. A subset S of $\boldsymbol{N}^k$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\boldsymbol{N}^k$ such that

$$S = \{v \mid v = v_0 + a_1 v_1 + \cdots + a_t v_t,\ a_i \in \boldsymbol{N}\}.$$

The vectors v_0 (referred to as the *constant vector*) and $v_1, v_2, \ldots, v_t$ (referred to as the *periods*) are called the *generators* of the linear set S. The set $S \subseteq \boldsymbol{N}^k$ is *semilinear* if it is a finite union of linear sets.

The empty set is a trivial (semi)linear set, where the set of generators is empty. Every finite subset of $\boldsymbol{N}^k$ is semilinear—it is a finite union of linear sets whose generators are constant vectors. It is also clear that the semilinear sets are closed under (finite) union.

Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. For each word w in Σ^*, define the *Parikh map* of w to be

$$\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n}).$$

For a language $L \subseteq \Sigma^*$, the *Parikh map* of L is $\psi(L) = \{\psi(w) \mid w \in L\}$. The language L is *semilinear* if $\psi(L)$ is a semilinear set.

Obviously, if a family $\mathscr{L}(= \mathscr{L}(\mathscr{M}))$ is effectively semilinear (that is, for each $M \in \mathscr{M}$, the semilinear set $S = \psi(L)$ can be effectively constructed), then the emptiness problem for $\mathscr{L}$ is decidable.

The following result was shown in [18].

THEOREM 3.1. *Let M be a one-way nondeterministic finite automaton* (*respectively, a one-way nondeterministic pushdown automaton*). *Then $\psi(L(M))$ is a semilinear set effectively computable from M.*

We can augment a one-way nondeterministic pushdown automaton with finitely many *reversal-bounded counters*, i.e., each counter can be tested for zero and can be incremented or decremented by one, but the number of alternations between nondecreasing mode and nonincreasing mode in any computation is bounded by a given constant. For example, a counter whose values change according to the pattern 0 1 1 2 3 4 4 3 2 1 0 1 1 0 is 3-reversal (here the reversals are underlined).

The next result, which generalizes Theorem 3.1, was proved in [13].

THEOREM 3.2. *Let M be a one-way nondeterministic finite automaton* (*respectively, a one-way nondeterministic pushdown automaton*) *augmented with finitely many reversal-bounded counters. Then $\psi(L(M))$ is a semilinear set effectively computable from M.*

There is a simple automata characterization of semilinear sets. Let M be a nondeterministic finite automaton *without an input tape*, but with n counters (for some $n \geqslant 0$). The computation of M starts with all the counters zero and the automaton in the start state. An atomic move of M consists of incrementing at most one counter by 1 and changing the state (decrements are not allowed). An n-tuple $v = (i_1, \ldots, i_n) \in \mathbf{N}^n$ is generated by M if M, when started from its initial configuration, halts with v as the contents of the counters. The set of all n-tuples generated by M is denoted by $G(M)$. We call this automaton a *finite-state generator*.

THEOREM 3.3. *Let $n \geqslant 0$. A subset $S \subseteq \mathbf{N}^n$ is semilinear if and only if it can be generated by a finite-state generator with n counters.*

Proof. From the definition of a semilinear set, it is straightforward to construct a finite-state generator M such that $G(M) = S$ for a given S. Conversely, suppose M is a finite-state generator with n counters. We construct a one-way nondeterministic finite automaton with n reversal-bounded counters M' which operates as follows. Given an input x, M' first simulates M by generating in n counters an n-tuple $(i_1, \ldots, i_n)$. Then M' checks and accepts if the input $x = a_1^{i_1} \ldots a_n^{i_n}$, where the a_i's are distinct symbols. Clearly, $\psi(L(M')) = G(M)$, and therefore, by Theorem 3.2, $G(M)$ is a semilinear set. ∎

For a class $\mathscr{M}$ of automata, let $\mathscr{M}_c$ be the class obtained by augmenting each automaton in $\mathscr{M}$ with finitely many reversal-bounded counters.

We generalize Theorem 3.2 as follows.

THEOREM 3.4. *Let $\mathscr{M}$ be a class of automata such that the family $\mathscr{L}(\mathscr{M})$ is effectively semilinear. Then the family $\mathscr{L}(\mathscr{M}_c)$ is effectively semilinear. Hence, the emptiness problem for $\mathscr{L}(\mathscr{M}_c)$ is decidable.*

Proof. The proof is a generalization of the proof of Theorem 3.2. Let $M_c \in \mathscr{M}_c$ with k reversal-bounded counters. We may assume, without loss of generality, that for inputs that are accepted, each counter starts and ends with zero value. Thus, each counter makes an odd number of reversals. In fact, we can assume that each counter makes exactly one reversal, since a counter that makes r reversals can be converted to $(r+1)/2$ counters, each making one reversal (one counter for each change from a nonincremental move to an incremental move). We show that $\psi(L(M_c))$ is an effectively computable semilinear set. We give the proof for $k = 1$, i.e., M_c has only one 1-reversal counter. The proof for the general case follows inductively from this.

Let $\Sigma = \{a_1, \ldots, a_n\}$ be the input alphabet of M_c, and $L_c = L(M_c)$ be the language accepted by M_c. Denote $\Sigma_1 = \Sigma \cup \{d, e\}$, where $d, e \notin \Sigma$, and define a morphism $\varphi : \Sigma_1^* \to \Sigma^*$ by $\varphi(a) = a$ for $a \in \Sigma$ and $\varphi(d) = \varepsilon = \varphi(e)$.

We construct an automaton M in $\mathscr{M}$ whose input alphabet is Σ_1 accepting a language $L \subseteq \Sigma^* \sqcup d^* e^*$. The automaton M works as follows on an input word $w \in \Sigma_1^*$. M simulates the computation of M_c on $w' = \varphi(w)$ and uses the letters d and e in w to simulate the actions of the counter: an increment "+1" (respectively, decrement "−1") corresponds to reading a symbol d (respectively, e) on the input.

Simultaneously, M checks that $w \in \Sigma^* \sqcup d^* e^*$. At some point during the simulation, M guesses that the number of e's read is equal to the number of d's (corresponding to the counter becoming zero); M continues the simulation, making sure that there are no more d's and e's encountered, and accepts if M_c accepts.

Therefore, $L \subseteq \Sigma^* \sqcup d^* e^*$, and for each $w' \in L_c$, there exists a word $w \in L$ such that $w \in \Sigma^* \sqcup \{d^i e^i \mid i \geqslant 0\}$ and $\varphi(w) = w'$.

By assumption, the Parikh map $\psi(L)$ of L is an effectively computable semilinear set S_1. Let S_2 be the semilinear set $\{(i_1, \ldots, i_n, k, k) \mid i_j, k \in \mathbf{N}\}$, where the last two coordinates correspond to symbols d and e, respectively. The set $S_3 = S_1 \cap S_2$ is semilinear (here intersecting with S_2 essentially gets rid of the "nonvalid computations"). The set S_3 is effectively computable, since semilinear sets are effectively closed under intersection [7]. Now the semilinear set S corresponding to the Parikh map $\psi(L_c)$ of L_c is obtained from S_3 by simply removing the last two coordinates of the tuples. ∎

We continue the proof of the previous theorem to show

THEOREM 3.5. *Let $\mathscr{M}$ and $\mathscr{D}$ be two classes of automata. If $\mathscr{L}(\mathscr{M}) = \mathscr{L}(\mathscr{D})$, then also $\mathscr{L}(\mathscr{M}_c) = \mathscr{L}(\mathscr{D}_c)$.*

Proof. Let $L_c = L(M_c) \subseteq \Sigma^*$ for $M_c \in \mathscr{M}_c$. We adopt the notations, conventions and the construction of the proof of Theorem 3.4. In particular, M_c has only one 1-reversal augmented counter. Let $K = \Sigma^* \sqcup \{d^n e^n \mid n \geqslant 0\}$, and let $L = L(M)$, where $M \in \mathscr{M}$ is the simulating automaton. Clearly, $L_c = \varphi(L \cap K)$, and, by assumption, there is an $M' \in \mathscr{D}$ such that $L = L(M')$. Hence $L_c = \varphi(L(M') \cap K)$. Now $L \times (M'_c) = L \cap K$ for some $M'_c \in \mathscr{D}_c$, since for the intersection with K we can use one new 1-reversal counter. Finally, $L(M''_c) = \varphi(L \cap K)$ for some $M''_c \in \mathscr{D}_c$, since for the morphic image w.r.t. φ, no new counters are needed. ∎

Note that Theorem 3.5 does not hold in converse. Indeed, for this it suffices to choose $\mathscr{M}$ as the class of all finite automata, and $\mathscr{D} = \mathscr{M}_c$.

A *simple shuffle language* is a language of the form $\Sigma^* \sqcup \{d^n e^n \mid n \geqslant 0\}$, for some alphabet Σ and distinct symbols d, e.

COROLLARY 3.1. *$\mathscr{L}(\mathscr{M}_c)$ is the smallest class of languages containing $\mathscr{L}(\mathscr{M})$ that is closed under morphisms and intersections with simple shuffle languages.*

4. EXAMPLES OF AUTOMATA CLASSES

We give some examples of classes of automata with reversal-bounded counters that accept only semilinear languages, and hence, their emptiness problem is decidable.

We first show that Theorem 3.2 is a corollary to Theorem 3.4.

- A *PCA* is a one-way nondeterministic pushdown automaton augmented with reversal-bounded counters.

By Theorem 3.1, every context-free language $L \in \mathscr{L}(PDA)$ is semilinear. Hence, by Theorem 3.4, the languages in $\mathscr{L}(PCA)$ are semilinear and, therefore their emptiness problem is decidable (this is Theorem 3.2).

- A *CA* is a one-way nondeterministic finite automaton augmented with reversal-bounded counters. Thus, it is a PCA without a pushdown stack.
- A 2*CA* is a two-way nondeterministic finite automaton with end markers augmented with reversal-bounded counters.

We note that the emptiness problem for 2CAs is undecidable, even when the automaton is deterministic and it has only two reversal-bounded counters [13].

- A *finite-crossing* 2*CA* is a 2CA where the number of times the input head crosses the boundary between any two adjacent cells of the input tape (including the end markers) is bounded by a given constant.

Every finite-crossing 2CA can effectively be converted to a (one-way) CA [9]. (However, the nondeterminism is essential here, see [15].) Therefore,

COROLLARY 4.1. *The languages accepted by finite-crossing* 2*CAs are semilinear, and hence their emptiness problem is decidable.*

In contrast to the undecidability for 2CAs with two reversal-bounded counters, the emptiness problem is decidable for deterministic 2CAs when there is only one reversal-bounded counter [15]. These automata can accept fairly complex languages. For example, such an automaton can recognize the language $L = \{0^k 1^n \mid k \text{ divides } n\}$ that is not semilinear. Thus, unlike finite-crossing reversal-bounded multicounter automata, these automata can accept languages whose Parikh maps are not semilinear. The nondeterministic case is open.

PROBLEM 4.1. *Is the emptiness problem decidable for nondeterministic* 2*CAs with only one reversal-bounded counter*?

We can extend a PCA by allowing multiple pushdown stacks.

- An *MPA* is an automaton that has multiple pushdown stacks, ordered by name, say $S_1, \ldots, S_m$, such that it can only read the topmost symbol of the *first* nonempty stack. Note that without this restriction, an MPA can simulate a Turing machine.
- An *MPCA* is an MPA augmented with multiple reversal-bounded counters.

A move of an MPCA M depends only on the current state, the input symbol (or ε), the status of each counter (zero or nonzero), and the topmost symbol of the first nonempty stack, say S_i; initially, the first stack is set to some starting top symbol Z_0 and all other stacks are empty. The action taken in a move consists of the input being consumed, each counter being updated $(+1, 0, -1)$, the topmost symbol of S_i being popped and a word (possibly empty) being pushed onto each stack, and the next state being entered. An MPA can be quite powerful. For example, the language $\{x^k \mid x \in \{a, b\}^*\}$, where k is a fixed integer, can be accepted by an MPA. However, it

was shown in [1] that MPAs accept semilinear languages only, and thus, by Theorem 3.4, we have the following result.

COROLLARY 4.2. *The languages in* $\mathscr{L}(MPCA)$ *are semilinear, and their emptiness problem is decidable.*

Corollary 4.2 was also observed recently in [6].

- A *TA* is a one-way nondeterministic finite automaton with a finite-crossing two-way read/write worktape, i.e., the number of times the head crosses the boundary between any two adjacent cells of the worktape is bounded by a constant, independent of the computation.
- A *TCA* is a TA with augmented reversal-bounded counters [14].

Note that in a TA there is no bound on how long the head of the worktape can remain on a cell.

We show in Corollary 4.3 that TCAs accept only semilinear languages. Note that if the worktape is not finite-crossing, then the automaton is equivalent to a Turing Machine.

EXAMPLE 4.1. A TCA can be quite powerful. For example, let $L_0 \subseteq \Sigma^*$ over the alphabet $\Sigma = \{a, b, c, d\}$ consist of all the words

$$x = x_1 a c^{i_1} b x_2 \ldots x_n a c^{i_n} b x_{n+1} \qquad \text{where } n \geqslant 0, \text{ and } x_i \notin \Sigma^* a \Sigma^* b \Sigma^*$$

such that

$$|x|_d = \sum_{j=1}^{n} c_j.$$

For instance, $dacbacaccbdd \in L_0$, but $ddacbacaccbdd \notin L_0$. There exists a TCA M that accepts the language

$$L = \{x \# x \mid x \in L_0\},$$

where $\# \notin \Sigma$. Here M has one counter and operates in the following manner. Given input $x \# y$, M copies x on the worktape and checks that $x = y$ and resets the worktape head to the left end of x. It computes the *sum* in its counter by looking at the worktape and whenever it sees an a, it first checks that there is a matching b to the right and that all symbols in-between are c's. It then moves left (to a), adding the length of the run of c's to the counter. The process is repeated until the whole word has been examined. So far, M crosses any boundary between two adjacent cells on the worktape at most 7 times. M then resets the worktape head to the left end of the tape and checks that the number of d's is equal to the *sum* in the counter. Thus, M is 9-crossing, although the worktape head makes an unbounded number of turns, i.e., it is not finite-turn. Note also that M does not rewrite the worktape and it is deterministic, and therefore the full power of TCAs is not needed to accept L.

We shall now proceed to prove the semilinearity of the language accepted by a TA. The proof consists of the two lemmas below, where we say that a TA M is *nonsitting*,

if in any computation of M the read/write head does not sit on any tape cell, i.e., it always moves left or right of a cell in every step.

LEMMA 4.1. *Let M_1 be a TA (i.e., without counters). We can effectively construct a TA M_2 such that $L(M_2) = L(M_1)$ and M_2 is nonsitting.*

Proof. Note that M_2 cannot just simulate a sitting step by a left (or right) move followed by a right (or left) move. This is because the read/write head can sit on a cell an unbounded number of steps, and this would make M_2 not finite-crossing.

What M_2 can do is to use a new "dummy" symbol, say $\#$. M_2 begins the simulation of M_1 by writing a finite-length sequence of $\#$'s on the worktape, the length being chosen nondeterministically. M_2 simulates M_1, but whenever M_1 writes a symbol on a *new* tape cell, M_2 also writes to the right of this cell a finite-length sequence of $\#$'s, the length of which is chosen nondeterministically. Thus, at any time, the worktape contains a word where every pair of nondummy symbols is separated by a word of $\#$'s. During the simulation, M_2 uses/moves on the $\#$'s to simulate the sitting moves of M_1, which is possible if there are enough $\#$'s between any pair of nondummy symbols. To simulate a nonsitting move of M_1, M_2 may need to "skip over" the $\#$'s to get to the correct nondummy symbol. Clearly, M_2 is nonsitting and accepts $L(M_1)$. ∎

LEMMA 4.2. *Let M be a TA. Then $\psi(L(M))$ is an effectively computable semilinear set.*

Proof. By Lemma 4.1, we assume that M is nonsitting. We may also assume the following: M's read/write worktape is one-way infinite and that a blank cell when visited is rewritten by a nonblank symbol, unless the automaton halts directly upon visiting the cell. We number the worktape cells by $1, 2, \ldots$ from left to right.

Consider an accepting computation of M on an input x such that M uses n worktape cells. By assumption, M accepts with its read/write head on cell n, which is blank. Now look at the cell p of the worktape, $p = 1, 2, \ldots, n$. In the computation, cell $p < n$ may be visited several times, but cell n is visited exactly once, on acceptance. Let $t_1, \ldots, t_m$ be the times M visits p.

Corresponding to the time sequence $(t_1, \ldots, t_m)$ associated with p, we define a *crossing vector* $R = (I_1, \ldots, I_m)$, where for each i, $I_i = (d_1, q_1, r_1, r_2, d_2)$,

- $d_1 \in \{-1, +1\}$ is the direction from which the head entered p at time t_i;
- q_1 is the state when M entered p;
- r_1 is the instruction that was used in the move above;
- r_2 is the instruction that was used at time $t_i + 1$ when M left p;
- $d_2 \in \{-1, +1\}$ is the direction to which M left p at time $t_i + 1$.

Note that instruction r_2 specifies the input $a_i \in \Sigma \cup \{\varepsilon\}$ that M reads at time $t_i + 1$. Denote $\gamma(R) = a_1 \ldots a_m$.

We construct a one-way nondeterministic finite-state automaton M' that simulates an accepting computation of M on w by nondeterministically guessing the sequence

of crossing vectors $R_1, \ldots, R_n$ as M' processes the worktape from left to right, making sure that R_j and R_{j+1} are compatible for $1 \leqslant j \leqslant n-1$. During the simulation, M' also checks that its input is $v = \gamma(R_1)\gamma(R_2)\ldots\gamma(R_{n-1})$. Clearly, v is a permutation of w, and $\psi(v) = \psi(w)$. Hence, by Theorem 3.1, $\psi(L(M)) = \psi(L(M'))$ is a semilinear set. ∎

By Theorem 3.4,

COROLLARY 4.3. *The languages in $\mathscr{L}(TCA)$ are semilinear, and their emptiness problem is decidable.*

- A *QA* (*restricted queue-automaton*) is a one-way nondeterministic finite automaton with a queue such that the number of alternations between nondeletion phase and noninsertion phase is bounded by a constant. A *nondeletion* (*noninsertion*, respectively) phase is a period of a computation consisting of insertions (deletions, respectively) and *no-changes*, where the queue is idle.
- A *QCA* is a QA augmented with reversal-bounded counters [14].

Note that a QCA can be simulated by a TCA. Hence,

COROLLARY 4.4. *The languages in $\mathscr{L}(QCA)$ are effectively semilinear, and their emptiness problem is decidable.*

As noted earlier, the restriction that the worktape in a TCA is finite-crossing is necessary, since, otherwise, the automaton becomes a Turing machine. In fact, even in a special case, the emptiness problem is undecidable. Now we restrict the worktape to operate like a pushdown stack which can push (i.e., write) but *cannot pop* (i.e., erase), but can enter the stack in a read-only mode. Moreover, once it enters the stack in a read-only mode, it can no longer push. There is no restriction on the number of times the stack head can cross the boundary between any two stack cells. This restricted worktape is called a *checking tape*. We call this automaton CCA. Such automata without counters have been studied in [8]. The emptiness problem for 2CAs is undecidable when the automaton is deterministic and it has only two reversal-bounded counters [13]. Therefore, we have

THEOREM 4.1. *The emptiness problem for CCAs is undecidable, even when restricted to deterministic automata with only two reversal-bounded counters.*

5. THE EQUIVALENCE PROBLEMS FOR SETS OF MAPS

Let $g, h : \Sigma^* \to \Delta^*$ be two mappings for the alphabets Σ and Δ. The *equality set* of g and h is defined to be

$$E(g, h) = \{w \in \Sigma^* \mid g(w) = h(w)\}.$$

The problem whether $E(g, h) = \{\varepsilon\}$ is undecidable for the equality sets of morphisms. This problem is known as the *Post Correspondence Problem*.

Let $\mathscr{F}$ be a family of mappings between word monoids, and let $\mathscr{L}$ be a language family. The *equivalence problem of maps from $\mathscr{F}$ on $\mathscr{L}$* is the problem whether $L \subseteq E(g,h)$ for $L \in \mathscr{L}$ and $g,h \in \mathscr{F}$.

The equivalence problem of morphisms is known to be decidable on context-free languages [5]. However, since the universe problem is undecidable for the context-free languages, also the problem whether $L = E(h,g)$ is undecidable for morphisms h,g and context-free languages L.

We can generalize the problem to the *multiple equivalence problem of $\mathscr{F}$ on $\mathscr{L}$*: given a language $L \in \mathscr{L}$, and a finite set (g_i,h_i) of pairs of maps $g_i,h_i \in \mathscr{F}$ for $i = 1,2,\ldots,k$, determine whether

$$L \subseteq \bigcup_{i=1}^{k} E(g_i,h_i).$$

THEOREM 5.1. *Let $\mathscr{M}$ be a class of automata such that the family $\mathscr{L}(\mathscr{M})$ is effectively semilinear. The multiple equivalence problem of morphisms is decidable on $\mathscr{L}(\mathscr{M})$.*

Proof. Consider an instance $L = L(M)$, (g_i,h_i), $i = 1,2,\ldots,k$, of the problem, where $M \in \mathscr{M}$, $L \subseteq \Sigma^*$, and $g_i,h_i : \Sigma^* \to \Delta_i^*$ for each i. Let

$$L' = \{w \in L \mid g_i(w) \neq h_i(w) \text{ for all } i\}.$$

Clearly, each word $w \in L$ satisfies $g_i(w) = h_i(w)$ for some i if and only if $L' = \emptyset$. We show that L' can be accepted by an automaton $M' \in \mathscr{M}_c$, i.e., by an automaton obtained from a machine in $\mathscr{M}$ by augmenting it with reversal-bounded counters, and, hence, by Theorem 3.4, the emptiness problem of the associated languages L' is decidable.

The automaton M' will use 2k counters $c_{11},c_{12},\ldots,c_{k1},c_{k2}$. On an input w, the counters c_{i1} and c_{i2} are used to find a discrepancy in $g_i(w)$ and $h_i(w)$, that is, a position p_i, where the words $g_i(w)$ and $h_i(w)$ differ from each other.

Now M' simulates M and at the same time applies, simultaneously for all $i = 1,2,\ldots,k$, the morphisms g_i and h_i on the input w (without recording the images $g_i(w)$ and $h_i(w)$) by using the counter c_{i1} (c_{i2}, respectively) to guess and store the position p_{i1} in $g_i(w)$ (p_{i2} in $h_i(w)$, respectively) where a discrepancy might occur. Therefore, either $g_i(w) = u_1 b_{i1} v_1$, where $|u_1| = p_{i1} - 1$, or $b_{i1} = \varepsilon$ and $p_{i1} > |g_i(w)|$; and similarly for h_i, either $h_i(w) = u_2 b_{i2} v_2$, where $|u_2| = p_{i2} - 1$, or $b_{i2} = \varepsilon$ and $p_{i2} > |h_i(w)|$. The automaton M' also records the symbols $b_{i1},b_{i2} \in \Delta_i \cup \{\varepsilon\}$, and continues the simulation of M until the input word w is consumed. So far, M' has just increased the counters, and has made no reversals. Now, M' checks that for all i, $b_{i1} \neq b_{i2}$, and if M accepts the word w, then M' verifies that for all $i, p_{i1} = p_{i2}$ by simultaneously decrementing the counters c_{i1} and c_{i2} and checking that they reach zero at the same time. Finally, M' accepts when all the counter checks succeed. Clearly, M' accepts L', and the claim follows. ∎

As a corollary to Theorem 5.1, we obtain

COROLLARY 5.1. *It is decidable for finitely many morphisms $h_1,\ldots,h_k$ and a language L accepted by an automaton in $\mathscr{M}$ whether for each word $w \in L$, $h_i(w) = h_j(w)$ for some $i \neq j$.*

In the proof of Theorem 5.1, the crucial point is that M' can store the positions p_{i1} of $g_i(w)$ and p_{i2} of $h_i(w)$, and the corresponding symbols in these positions, simultaneously for all indexes i. Therefore, the proof generalizes to the case of mappings computed by deterministic generalized sequential machines (gsms) with reversal-bounded counters (and with or without accepting states). The following result improves a result of [5].

THEOREM 5.2. *Let $\mathscr{M}$ be a class of automata such that the family $\mathscr{L}(\mathscr{M})$ is effectively semilinear. The multiple equivalence problem of the mappings computed by the deterministic gsms with reversal-bounded counters is decidable on $\mathscr{L}(\mathscr{M})$.*

In particular,

COROLLARY 5.2. *It is decidable for finitely many mappings $h_1, h_2, \ldots, h_k$ computed by deterministic gsms and a language L accepted by an automaton in $\mathscr{M}$ whether for each word $w \in L$, $h_i(w) = h_j(w)$ for some $i \neq j$.*

The problem of whether for a given context-free language L and two morphisms h_1, h_2, $h_1(w) = (h_2(w))^{mi}$ holds for all $w \in L$, where the operation *mi* takes the mirror images, was shown to be decidable in [12]. We can generalize the multiple equivalence problem to *multiple mirror equivalence problem* by specifying that for certain indices i, we require that $g_i(w) = (h_i(w))^{mi}$ and for the rest that $g_j(w) = h_j(w)$. So, for example, if there are 3 indices, 1,2,3, we might want to decide whether, for each word $w \in L$, $g_1(w) = (h_1(w))^{mi}$ or $g_2(w) = h_2(w)$ or $g_3(w) = (h_3(w))^{mi}$.

THEOREM 5.3. *Let $\mathscr{M}$ be a class of automata such that the family $\mathscr{L}(\mathscr{M})$ is effectively semilinear. The multiple mirror equivalence problem is decidable for deterministic gsm mappings and languages accepted by automata in $\mathscr{M}$.*

In Theorem 5.3, if instead of checking $g_i(w) \neq h_i(w)$, we want to check that $g_i(w) \neq (h_i(w))^{mi}$, a similar construction as in the above works. Now M' applies h_i backwards on the input, and the p_{i2}, that is stored in counter c_{i2}, is measured from somewhere in the input word to the end of the input. If h_i is a deterministic gsm mapping, then the gsm is run backwards.

There are natural variations of the above theorems, e.g., one might be interested in deciding the problem, for given $L, h_1, \ldots, h_k$, and $m \leqslant k$, whether or not for each $w \in L$, there are at least m mappings h_i that map w to the same word.

PROBLEM 5.2. *For which language families is the following problem decidable? Let L be a language, h be a nondeterministic gsm mapping and g be a deterministic gsm mapping. Is there a word $w \in L$ such that $g(w) \notin h(w)$?*

Note that the above problem is known to be undecidable when both h and g are nondeterministic gsm mappings. In fact, quite surprisingly, the problem is undecidable even when both h and g are finite substitutions and the language L is fixed to be the simple bounded language ab^*c [16].

6. PROBLEMS ON COMMUTATION

In the *commutation problem* for a language family $\mathscr{L}$ we ask whether $LK = KL$ for two given languages $L, K \in \mathscr{L}$. In the *equivalence problem* for a family $\mathscr{L}$ we ask for given languages $L, K \in \mathscr{L}$ whether $L = K$.

THEOREM 6.1. *Let $\mathscr{L}$ be a language family containing the singleton sets and effectively closed under concatenation. Then the commutation problem is decidable for $\mathscr{L}$ if and only if the equivalence problem is decidable for $\mathscr{L}$.*

Proof. By the assumption, if $L, K \in \mathscr{L}$, then also $LK, KL \in \mathscr{L}$, and therefore if the equivalence problem is decidable for $\mathscr{L}$, so is the commutation problem, $LK = KL$.

On the other hand, assume that the commutation problem is decidable for $\mathscr{L}$. Let $L, K \in \mathscr{L}$, and let $\#$ be a symbol not in the alphabet of L and K. Then the equivalence $L = K$ can be decided, since $(L\#)(K\#) = (K\#)(L\#)$ if and only if $L = K$. ∎

It follows that the commutation problem is undecidable for context-free languages. In fact, it is undecidable for languages accepted by nondeterministic finite automata augmented with a 1-reversal counter, since for these automata, even with only 4 states the universe problem is undecidable, and therefore also their equivalence problem is undecidable [10].

Let $\mathscr{L}(DPDA)$ be the family of deterministic context-free languages, that is, those languages accepted by deterministic PDAs. The equivalence problem is decidable for the languages in $\mathscr{L}(DPDA)$ [20] (see also [21]), but $\mathscr{L}(DPDA)$ is not closed under concatenation. In fact, $\mathscr{L}(DPDA)$ is not closed under concatenation from the left by two element sets, see [11]. However, $\mathscr{L}(DPDA)$ is effectively closed under *marked concatenation*, that is, if $L, K \in \mathscr{L}(DPDA)$, then also $L\#K \in \mathscr{L}(DPDA)$, where $\#$ is a symbol not in the alphabets of L and K. Therefore, by the second part of the proof of Theorem 6.1, if the commutation problem turns out to be decidable for deterministic context-free languages, a direct proof of this is likely to be very difficult.

PROBLEM 6.3. *Is the commutation problem decidable for deterministic context-free languages*?

PROBLEM 6.4. *For which language families is the commutation problem, $LK = KL$, decidable when K is a finite language*?

A *deterministic finite-turn* 2*CA* is a special case of deterministic finite-crossing 2CA, where the head makes at most a fixed number of (left-to-right or right-to-left) turns on the input tape. The equivalence problem for these automata, in fact, for deterministic finite-crossing 2CAs, is decidable [9, 13, 15]. Moreover, it can be shown that, given a finite set K and a language L accepted by a deterministic finite-turn 2CA, we can effectively construct deterministic finite-turn 2CAs accepting KL and LK. It follows that Problem 6.4 is decidable for deterministic finite-turn 2CAs.

We shall now show that Problem 6.4 has a negative answer for context-free languages. Indeed, we prove that the commutation problem is undecidable for $K = \{a, b\}$ and context-free languages L. For the proof we need the following result from [4].

LEMMA 6.1. *Let K be a two-element code and L be any language. Then $LK = KL$ if and only if there exists a subset $I \subseteq \boldsymbol{N}$ such that*

$$L = \bigcup_{i \in I} K^i. \tag{1}$$

We note that if (1) holds for a code K and a language L, then the set I is uniquely determined, since $K^i \cap K^j = \emptyset$ for all $i \neq j$.

THEOREM 6.2. *Let K be a fixed two-element code. It is undecidable whether for a context-free language L there exists a set I such that $L = \bigcup_{i \in I} K^i$.*

Proof. Let $K = \{x_1, x_2\} \subseteq \Sigma^*$. Let $\Delta = \{a_1, a_2\}$ be an alphabet, and define a bijective morphism $\varphi : \Delta^* \to \Sigma^*$ by $\varphi(a_i) = x_i$ for $i = 1, 2$.

We first show that it is undecidable for context-free languages $L \subseteq \Sigma^*$ whether or not $L = K^*$. Indeed, for any context-free language L', $L' = \Delta^*$ if and only if $\varphi(L') = K^*$, and the claim follows, since $L = \varphi(L')$ is effectively context-free, and the universe problem is undecidable for context-free languages.

Suppose contrary to the claim of the theorem that the existence of a set I of powers can be decided. We derive a contradiction from the undecidability of the equivalence problem $L = K^*$. Let $L \subseteq \Sigma^*$ be any context-free language. If there does not exist a set I such that $L = \bigcup_{i \in I} K^i$, then trivially $L \neq K^*$. Suppose then that such an I exists. We show that I is a semilinear subset of $\boldsymbol{N}$ (that is, I is ultimately periodic) and it can be effectively constructed from L. We have $\varphi(\varphi^{-1}(L)) = L$, since $L \subseteq K^*$, and hence $\varphi^{-1}(L) = \bigcup_{i \in I} \Delta^i$. Because $\varphi^{-1}(L)$ is effectively context-free, the length set I of $\varphi^{-1}(L)$ is semilinear, and it can be effectively constructed from $\varphi^{-1}(L)$, and thus from L. It follows that L is effectively regular, and therefore we can decide whether or not $L = K^*$. This contradicts the undecidability claim in the beginning of the proof. ∎

By Lemma 6.1, we have the following corollary.

COROLLARY 6.1. *Let K be a fixed two element code. It is undecidable for context-free languages L whether or not $KL = LK$.*

Theorem 6.2 suggests the following general problem.

PROBLEM 6.5. *For which language families $\mathscr{L}$ is the following problem decidable: Given a (possibly infinite) regular code R and a language $L \in \mathscr{L}$, does there exist a set I such that $L = \bigcup_{i \in I} R^i$?*

By Theorem 6.2, Problem 6.5 is undecidable for context-free languages even in very simple cases of R. We show now that the problem is decidable for deterministic context-free languages.

If a set R is a code, then $R^i \cap R^j = \emptyset$ for all $i \neq j$, and therefore we have

LEMMA 6.2. *Let $R \subseteq \Sigma^*$ be regular code and $L \subseteq \Sigma^*$ a language. Then there exists a set I such that $L = \bigcup_{i \in I} R^i$ if and only if $L \subseteq R^*$ and, for all $i \geqslant 0$,*

$$L \cap R^i \neq \emptyset \Rightarrow \bar{L} \cap R^i = \emptyset. \tag{2}$$

THEOREM 6.3. *It is decidable for deterministic context-free languages L and regular codes R whether or not $L = \bigcup_{i \in I} R^i$ for some set I.*

Proof. The containment problem $L \subseteq R^*$ is decidable for context-free languages, since R^* is a regular language. Assume thus that $L \subseteq R^*$ holds. For (2), we recall that the deterministic context-free languages are effectively closed under complementation. Let $L = L(M_1)$ and $\bar{L} = L(M_2)$ for the deterministic pushdown automata M_1 and M_2, and let A be a finite automaton with $L(A) = R^*$. Let $\#$ be a new symbol.

We construct a nondeterministic pushdown automaton M augmented with one 1-reversal counter that accepts a word $u \# v \in \Sigma^* \# \Sigma^*$ if and only if $u \in L \cap R^i$ and $v \in \bar{L} \cap R^i$ for some i. Therefore $L(M) = \emptyset$ if and only if L satisfies condition (2) of the claim. Since the emptiness problem is decidable for the languages accepted by the pushdown automata augmented with reversal-bounded counters, the claim follows.

Let then $w = u \# v$ be an input word. Then M simulates M_1 and A in parallel on u, and M checks that $u \in L$ and $u \in R^i$ for some i, recording i in the counter. Note that i is unique, because R is a code. In this part M needs to be nondeterministic. Then M simulates M_2 and A on v and checks that $v \in \bar{L}$ and $v \in R^i$ for the same i that was recorded in the counter. This M does by decrementing the counter. Finally, M accepts if M_2 accepts. ∎

Theorem 6.3 generalizes in many different ways to larger language families. We note that condition (2) is decidable also in the following cases (for the definitions see Section 4):

- L is accepted by a deterministic MPCA and R is accepted by a CA.
- L is accepted by a deterministic CA and R is accepted by an MPCA.
- L is accepted by deterministic finite-crossing 2CA (since the class of languages defined is effectively closed under complementation [9]) and R is accepted by a CA.
- L and R are accepted by deterministic 2CAs with only one reversal-bounded counter, as this class is effectively closed under complementation [15].

However, condition (2) is not decidable when R is a deterministic context-free code. In fact, we have a general undecidability result in Theorem 6.4. The *disjointness problem* for a language family $\mathscr{L}$ is the problem whether $L_1 \cap L_2 = \emptyset$ for two languages $L_1, L_2 \in \mathscr{L}$.

THEOREM 6.4. *Let $\mathscr{L}$ be a language family that is effectively closed under concatenation and union with singleton sets. If the disjointness problem is undecidable for $\mathscr{L}$, then so is condition* (2) *for codes $L, R \in \mathscr{L}$.*

Proof. Given $L_1, L_2 \in \mathscr{L}$ over the alphabet Σ, define L and R over the alphabet $\Sigma \cup \{\#, \$\}$ by: $L = L_1\#$ and $R = L_2\# \cup \{\$\}$. Clearly, both L and R are codes in $\mathscr{L}$. Now, there exists an i such that $L \cap R^i \neq \emptyset$ and $\bar{L} \cap R^i \neq \emptyset$ if and only if $L_1 \cap L_2 \neq \emptyset$. Hence, the undecidability follows. ∎

In particular, Theorem 6.4 holds for the following cases:

- L and R are accepted by deterministic pushdown automaton whose stack is 1-turn, i.e., after the stack has popped, it can no longer push.
- L and R are accepted by deterministic one-counter automata, i.e., each automaton has one unrestricted counter.

Indeed, the disjointness problem is undecidable for the language family accepted by deterministic 1-turn pushdown automata and deterministic one-counter automata, respectively, see, e.g., [13].

We end this section with a problem for reversals of languages. Recall that L^{mi} denotes the mirror language of L. The language L is *mirror closed*, if $L = L^{mi}$. In the *mirror closure problem* for $\mathscr{L}$ we ask whether $L = L^{mi}$ for a given language $L \in \mathscr{L}$.

THEOREM 6.5. *Let $\mathscr{L}$ be a language family effectively closed under taking mirror images and marked concatenation. Then the mirror closure problem is decidable for $\mathscr{L}$ if and only if the equivalence problem is decidable for $\mathscr{L}$.*

Proof. If the equivalence problem is decidable, then, by the assumption on $\mathscr{L}$, so is the mirror closure problem, $L = L^{mi}$. In the other direction, the claim follows from the equivalence: $L\# K^{mi} = (L \# K^{mi})^{mi}$ if and only if $L = K$. ∎

It follows, e.g., that the mirror closure problem is undecidable for context-free languages, and decidable for regular languages.

PROBLEM 6.6. *For which language families is the mirror closure problem decidable*?

Problem 6.6 is decidable for languages accepted by deterministic finite-crossing 2CAs, since this class is effectively closed under taking mirror images and its equivalence problem is decidable [9, 13, 15].

REFERENCES

1. L. Breveglieri, A. Cherubini, C. Citrini, and S. Crespi Reghizzi, Multiple pushdown languages and grammars, *Internat. J. Found. Comput. Sci.* **7** (1996), 253–291.
2. C. Choffrut and J. Karhumäki, Combinatorics of words, *in* "Handbook of Formal Languages," (A. Salomaa and G. Rozenberg, Eds.), Vol. 1, pp. 329–438, Springer-Verlag, Berlin, 1997.

3. C. Choffrut and J. Karhumäki, Characterizing the subsets of words commuting with a set of two words, *in* "Proceedings of the 7th Nordic Combinatorial Conference," T. Harju and T. Honkala, Eds.), Turku, Finland, 1999.
4. C. Choffrut, J. Karhumäki and N. Ollinger, The commutation of finite sets: Challenging problem, *Theoret. Comput. Sci.* (2001) to appear.
5. K. Culik II and A. Salomaa, On the decidability of homomorphism equivalence for languages, *J. Comput. System Sci.* **17** (1978), 163–175.
6. Z. Dang, "Verification and Debugging of Infinite State Real-Time Systems," Ph.D. thesis, University of California, Santa Barbara, 2000.
7. S. Ginsburg, "The Mathematical Theory of Context-Free Languages," McGraw–Hill, New York, 1966.
8. S. A. Greibach, Checking automata and one-way stack languages, *SDC Document TM 738/045/00*, 1968.
9. E. M. Gurari and O. H. Ibarra, The complexity of decision problems for finite-turn multicounter machines, *J. Comput. System Sci.* **22** (1981), 220–229.
10. V. Halava and T. Harju, Undecidability in integer weighted finite automata. *Fundamenta. Inf.* **38** (1999), 189–200.
11. M. Harrison, "Introduction to Formal Language Theory," Addison–Wesley, Reading, MA, 1978.
12. S. Horvath, J. Karhumäki, and H. C. M. Kleijn, Results concerning palindromicity, *J. Internat. Process. Cyber. EIK* **23** (1987), 441–451.
13. O. H. Ibarra, Reversal-bounded multicounter machines and their decision problems, *J. ACM* **25** (1978), 116–133.
14. O. H. Ibarra, T. Bultan, and J. Su, Reachability analysis for some models of infinite-state transition systems, *in* "Proceedings of the 11th International Conference on Concurrency Theory (CONCUR 2000)," (Catuscia Palamidessi, Ed.), *Lecture Notes in Comput. Sci.* Vol. 1877, pp. 183–198, Springer-Verlag, Berlin, 2000.
15. O. H. Ibarra, T. Jiang, N. Tran, and H. Wang, New decidability results concerning two-way counter machines, *SIAM J. Comput.* **24** (1995), 123–137.
16. J. Karhumäki, and L. P. Lisovik, A simple undecidable problem: the inclusion problem for finite substitution on ab^*c, *Inform. and Comput*. (2001), to appear.
17. M. Lothaire, "Combinatorics on Words," Addison–Wesley, Reading, MA, 1983.
18. R. Parikh, On context-free languages, *J. ACM* **13** (1966), 570–581.
19. A. Salomaa, "Formal Languages," Academic Press, New York, 1973.
20. G. Sénizergues, The equivalence problem for deterministic pushdown automata is decidable, *Theoret. Comput. Sci.* **251** (2001), 1–166.
21. C. Stirling, Decidability of DPDA equivalence, *Theoret. Comput. Sci.* **225** (2001), 1–31.