Automata with Reversal-Bounded Counters: A Survey

Oscar H. Ibarra

Department of Computer Science,
University of California, Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu

Abstract. We survey the properties of automata augmented with reversal-bounded counters. In particular, we discuss the closure/non-closure properties of the languages accepted by these machines as well as the decidability/undecidability of decision problems concerning these devices. We also give applications to several problems in automata theory and formal languages.

Keywords: finite automaton, pushdown automaton, visibly pushdown automaton, transducer, context-free grammar, reversal-bounded counters, semilinear set, finitely-ambiguous, finite-valued, decidable, undecidable.

1 Introduction

A counter is an integer variable that can be incremented by 1, decremented by 1, and tested for zero. It starts at zero and can only store nonnegative integer values. Thus, one can think of a counter as a pushdown stack with a unary alphabet, in addition to the bottom of the stack symbol which is never altered.

An automaton (DFA, NFA, DPDA, NPDA, etc.) can be augmented with multiple counters, where the "move" of the machine also now depends on the status (zero or non-zero) of the counters, and the move can update the counters. See [27] for formal definitions.

It is well known that a DFA augmented with two counters is equivalent to a Turing machine (TM) [39]. However, when we restrict the operation of the counters so that during the computation, the number of times each counter alternates between nondecreasing mode and nonincreasing mode is at most some fixed number r (such a counter is called *reversal-bounded*), the computational power of a DFA (and even an NPDA) augmented with reversal-bounded counters is significantly weaker than a TM. When $r = 1$, we refer to the counters as 1-reversal (thus, once a counter decrements, it can no longer incerement). Note that a counter that makes r reversals can be simulated by $\lceil \frac{r+1}{2} \rceil$ 1-reversal counters.

A k-NCM (resp., k-DCM, k-NPCM, k-DPCM) is an NFA (resp., DFA, NPDA, DPDA) augmented with k reversal-bounded counters. For machines with two-way input head operating on an input with left and right end markers, we use the notation k-2NCM, k-2NPCM, etc. When the number of counters is not specified, we just write NCM, NPCM, etc. So, e.g., a 2NPCM is a two-way NPDA augmented with some number of reversal-bounded counters.

Automata with reversal-bounded counters were first studied in [4,27], where closure and decision properties were investigated. For example, in [27], it was shown that the class of languages accepted by DCMs is effectively closed under union, intersection, and complementation (closure under the first two operations also hold for NCMs), and emptiness, finiteness, disjointness, containment, and equivalence problems are decidable.

NCMs and NPCMs and their two-way versions have been extensively studied. Many generalizations have been introduced, see, e.g., [15,5,7,8]. They have found applications in areas like timed-automata [12,10,11,6], transducers [21,23,2,3,16,38,29], membrane computing [33], DNA computing [30], verification [12,37,31,11,13,46,5,24], and Diophantine equations [22,14,32,45].

Here, we give a survey of important results concerning machines augmented with reversal-bounded counters and summarize some recent developments involving the use of these machines for solving problems in automata theory and formal languages.

2 Decidability of the Emptiness and Infiniteness Problems

Let $\mathbb{N}$ be the set of nonnegative integers and m be a positive integer. A subset Q of $\mathbb{N}^m$ is a *linear set* if there exist vectors $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{N}^m$ such that $Q = \{\boldsymbol{v}_0 + i_1\boldsymbol{v}_1 + \cdots i_n\boldsymbol{v}_n \mid i_1, \ldots, i_n \in \mathbb{N}\}$. The vectors $\boldsymbol{v}_0$ (referred to as the *constant*) and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ (referred to as *periods*) are called the *generators* of the linear set Q. A finite union of linear sets is called a *semilinear set.* Every finite subset of $\mathbb{N}^m$ (including the empty set $\varnothing$) is semilinear – it is just a finite union of linear sets with no periods.

Let $\Sigma = \{a_1, \ldots, a_m\}$. For a word w over Σ and a letter $a \in \Sigma$, we denote by $|w|_a$ the number of occurrences of a's in w. The *Parikh map* of w is the m-dimensional vector $\phi(w) = (|w|_{a_1}, \ldots, |w|_{a_m})$. The *Parikh map* of a language $L \subseteq \Sigma^*$ is defined as $\phi(L) = \{\phi(w) \mid w \in L\}$. (The reader is assumed to be familiar with basic notions in formal languages and automata theory; see, e.g., [26].)

The following fundamental result was first shown in 1978 [27]:

Theorem 1. *The emptiness and infiniteness problems for NPCMs are decidable.*

Proof. We briefly sketch the proof given in [27]. Let M be a k-NPCM with input alphabet $\Sigma = \{a_1, \ldots, a_m\}$. We may assume that each counter is 1-reversal and acceptance is when the machine enters an accepting state when all the counters are zero. We construct an NPDA (without counters) M' over alphabet $\Sigma \cup \Delta$, where $\Delta = \{(i, +1), (i, -1) \mid 1 \leq i \leq k\}$. The symbol $(i, +1)$ (resp. $(i, -1)$) encodes the possible action of counter i in one step of the computation, i.e., encountering $(i, +1)$ (resp., $(i, -1)$) on the input represents incrementing (resp., decrementing) counter i. If w is an input to M, then an input to M' would be a string w' which would consist of w shuffled with some string z in Δ^*, where for each i, all the $(i,+1)$'s appear before all the $(i,-1)$'s. M' simulates M and whenever counter i increments (resp., decrements), M' checks that its input head is on symbol $(i,+1)$ (resp., $(i,-1)$) before moving right. M' accepts w' if M accepts w. Since M is an NPDA, the Parikh map of $L(M')$ is an effectively computable semilinear set $Q_1 \subseteq \mathbb{N}^{m+2k}$, where the first m coordinates correspond to the symbols in Σ and the last $2k$ coordinates correspond to the symbols in Δ (in the order: all the $(i,+1)$'s followed by all the $(i,-1)$'s). Let $Q_2 = \{(i_1,\ldots,i_m,j_1,\ldots,j_k,j_1,\ldots,j_k) \mid i_r \geq 0 \text{ for } 1 \leq r \leq m,\ j_s \geq 0 \text{ for } 1 \leq s \leq k\}$. Clearly, Q_2 is semilinear. Hence, $Q_3 = Q_1 \cap Q_2$ is semilinear, since semilinear sets are closed under intersection. Semilinear sets are also closed under projection; hence, we can obtain the semilinear set $Q \subseteq \mathbb{N}^m$ corresponding to the Parikh map of $L(M)$ by deleting the last $2k$ coordinates in Q_3. The result follows, since the emptiness and infiniteness problems for semilinear sets are decidable [18]. □

When the number k of counters and the bound r on the counter reversals are *fixed*, an upper bound of NPTIME for the nonemptiness decision procedure was shown by Filiot et al at MFCS 2010 [16]. Their proof consisted of a careful analysis of the procedure in [27] (which is briefly described in the proof above) for showing that the Parikh map of the language accepted by an NPCM M whose counters are 1-reversal counters is semilinear and then appealing to two results in [41]: constructing an existential Presburger formula for representing the Parikh map of the language accepted by the NPCM M can be accomplished in $O(|M|)$ time, and satisfiability of existential Presburger formula is in NPTIME.

In the above improvement, k and r are assumed to be fixed; otherwise, the procedure would be in NEXP. However, recently, the complexity of the emptiness problem has been improved. In their CAV 2012 paper, Hague et al [24] showed that the emptiness problem for NPCMs (thus the number of counters and the bound r on reversals are not fixed, but r is assumed to be written in unary) is NP-complete, even when the counters can be compared against and incremented/decremented by constants that are given in binary. The construction in [24] used a direct polynomial-time reduction to satisfaction over existential Presburger formulas.

For the case of NCMs, the following was shown in [20]:

Theorem 2. *For fixed k and r, the emptiness problem for k-NCMs whose counters are r-reversal is in NLOGSPACE (hence, also in PTIME).*

For the machine models studied in the rest of the paper, we will just show the decidability of emptiness and infiniteness, and not discuss their complexity. However, one should be able to obtain lower and upper bounds on the complexity of the emptiness problems using the results and techniques in [20,24].

A 2NCM is finite-crossing if there is an integer c such that the input head crosses the boundary between any two adjacent input symbols at most c times. The following was shown in [20]:

Theorem 3. *Every finite-crossing 2NCM can effectively be converted to an equivalent NCM. Hence, the emptiness and infiniteness problems for finite-crossing 2NCMs are also decidable.*

Finite-crossing 2NCMs are quite powerful. For example, they can accept languages that are not context-free. However, from Theorem 3, every NCM can be converted to an NCM (i.e., one-way). By using the result of Baker and Book [4] that every NCM can converted to one that runs in linear time, it is easy to show that the context-free language $L = \{x\#x^r \mid x \in (a+b)^+\}$ cannot be accepted by an NCM and, hence, cannot be accepted by a finite-crossing 2NCM.

Theorems 1 and 3 can be generalized. Define a 3-phase finite-crossing 2NPCM M which operates in three phases: In the first phase, M operates as a finite-crossing 2NCM without using the stack. In the second phase, with the configuration (state, input head position, and counter values) the first phase left off, M operates as an NPCM where the head can only move right on the input. Finally, in the third phase with the configuration (state, head position, counter values but not the stack) the second phase left off, M operates again as a finite-crossing 2NCM without using the stack. It is possible that the machine has only one or two phases. So, e.g., M can accept with only Phase 1, or with only Phases 1 and 2. A 3-phase finite-crossing 2DPCM is one in which all phases are deterministic. The following was shown in [38]:

Theorem 4. *The emptiness and infiniteness problems for 3-phase finite-crossing 2NPCMs are decidable.*

Theorem 4 seems to be the strongest result that we can prove in the sense that we cannot generalize the NPCM in the second phase to be a finite-crossing 2NPCM. In fact, it can be shown that a 2DPDA which makes only 3 reversals on the stack and 2-turns on the input has an undecidable emptiness problem.

An automaton M is over a letter-bounded language (resp., word-bounded language) if $L(M) \subseteq a_1^* \cdots a_k^*$ for some $k \geq 1$ and distinct letters $a_1, \ldots, a_k$ (resp., $L(M) \subseteq w_1^* \cdots w_k^*$ for some $k \geq 1$ and not-necessarily distinct words $w_1, \ldots, w_k$).

The following result appeared in [38]:

Theorem 5. *The emptiness and infiniteness problems for finite-crossing 2DPCMs over word-bounded languages are decidable.*

When we remove the "finite-crossing" condition in Theorem 3, we get the following:

Theorem 6. 1. *The emptiness problem for 2DFAs (even over letter-bounded languages) with two reversal-bounded counters is undecidable. [27].*
2. *The emptiness problem for 2DFAs with one reversal-bounded counter is decidable [34].*
3. *The emptiness problem for 2NFAs with one reversal-bounded counter over word-bounded languages is decidable [14].*

2DFAs and 2NFAs augmented with one reversal-bounded counter are rather powerful. For example the non-semilinear language $L = \{a^{mi}b^i \mid i, m \geq 1\}$ can be accepted by a 2DFA with one counter that only reverses once. Applications of the decidability of the emptiness problem for these machine (over word-bounded languages for the nondeterministic case) to decision problems in verification and Diophantine equations, etc. can be found in [22,32,45].

Open: Is emptiness decidable for 2NFAs with one reversal-bounded counters (i.e., the inputs are unrestricted)?

Theorem 3 does not hold for 2DPDAs, even when they have no reversal-bounded counters and the input head only makes 1 turn on the input: a left-to-right sweep of the input followed by a right-to-left sweep. In fact, we can even restrict the stack to make only one reversal. Consider, e.g., the language $L = \{a^1\#a^2\#a^3\# \cdots \#a^{n-1}\#a^n\# \mid n \geq 1\}$. As shown in [36], one can construct a 2DPDA M which makes only 1-turn on the input and one reversal on the stack to accept L. This language can be accepted also by a 2DCA (two-way deterministic one-counter automaton) which makes only 1-turn on the input.

3 VPDAs with Reversal-Bounded Counters

The model of a visibly pushdown automaton (VPDA) was first introduced and studied in [1]. It is an NPDA where the input symbol determines the (push/stack) operation of the stack. The input alphabet Σ is partitioned into three disjoint alphabets: $\Sigma_c, \Sigma_r, \Sigma_{int}$. The machine pushes a specified symbol on the stack if it reads a call symbol in Σ_c on the input; it pops a specified symbol (if the specified symbol is at top of the stack) if it reads a return symbol in Σ_r on the input; it does not use the (top symbol of) the stack and can only change state if it reads an internal symbol in Σ_{int} on the input. The machine has no ε-moves, i.e, it reads an input at every step. An input $x \in \Sigma^*$ is accepted if the machine, starting from one of a designated set of initial states with a distinguished symbol $\perp$ at bottom of the stack (which is never altered), eventually enters an accepting sate after processing all symbols in x. For details, see [1]. In this paper, we assume without loss of generality, that the VPDA has only one initial state.

Recall that a 1-ambiguous NPDA is one where every input is accepted in at most one accepting computation. A 1-ambiguous NPDA is more powerful than a visibly pushdown automaton (VPDA) since the former can accept languages, like $L_1 = \{x\#x^R \mid x \in (a+b)^+\}$ and $L_2 = \{a^kba^k \mid k \geq 1\}$, that cannot be accepted by the latter. There are languages accepted by 1-ambiguous NPDAs that are not deterministic context-free languages (DCFLs), whereas languages accepted by VPDAs are DCFLs [1]. Note also that a 1-ambiguous NPDA can have ε-moves.

Now consider a VPDA augmented with k reversal-bounded counters. We allow the machine to have ε-moves, but in such moves, the stack is not used, only the state and counters are used and updated. Acceptance of an input string is when machine eventually falls off the right end of the input in an accepting state. Thus, a VPDA M with k reversal-bounded counters operates like a VPDA but can now use reversal-bounded counters as auxiliary memory.

Consider the language $L_1 = \{w \mid w = xy$ for some $x \in (a+b)^+, y \in (0+1)^+, x$ when a, b are mapped to $0, 1$, respectively, is the reverse of $y\}$. Clearly, L_1 can be accepted by a VPDA M_1. However, the language $L_2 = \{w \mid w \in L_1$, the number of a's + number of 0's in w = the number of b's + number of 1's in $w\}$ cannot be accepted by a VPDA. But L_2 can be accepted by a VPDA M_2 with two 1-reversal counters C_1 and C_2 as follows: On a given input w, M_2 simulates M_1, and stores the number of a's and 0's (resp., the number of b's and 1's) it sees on the input in counters C_1 and C_2, respectively. Then when M_1 accepts, M_2 (on ε-moves) decreases the counters simultaneously and accepts if the counters become zero at the same time.

Denote a VPDA with k reversal-bounded counters by k-VPCM, and by VPCM if the number of reversal-bounded counters is not specified. Clearly, VPCMs are a special case of NPCMs. Hence, from Theorem 1, we have:

Theorem 7. *The emptiness and infiniteness problems for VPCMs are decidable.*

Theorem 8. *The class of languages accepted by VPCMs is closed under union, intersection, renaming, concatenation, Kleene-*.*

Proof. The constructions are similar to the those for VPDAs without reversal-bounded counters in [1]. For example, for intersection, suppose M_1 and M_2 are two VPCMs with k_1 and k_2 reversal-bounded counters. We construct a VPCM with $k_1 + k_2$ reversal-bounded counters to simulate M_1 and M_2 in parallel. However, since the machines can have ε-transitions (which do not involve the stack), whenever one machine, say M_1, wants to make an ε-move but the other machine M_2 wants to make a non-ε-move, the simulation of M_2 is suspended temporarily until M_1 decides to make a non-ε-move, at which time the parallel simulation of both machine can be resumed. □

With respect to complementation, unlike for VPDAs, we have:

Theorem 9. *The class of languages accepted by VPCMs is not closed under complementation.*

Proof. Consider the language $L = \{x\#y \mid x, y \in (a+b)^+, x \neq y\}$. Clearly, L can be accepted by a VPCM (where $a, b, \#$ are internal inputs) that does not use the stack, but uses a reversal-bounded counter to verify if an input string is in L. Suppose the complement, L^c, of L, can be accepted by a VPCM. Let $L_1 = (a+b)^+\#(a+b)^+$. Since L_1 is regular it can be accepted by a VPDA. Then, by Theorem 8, $L_2 = L^c \cap L_1 = \{x\#x^R \mid x \in (a+b)^+\}$ can be accepted by a VPCM M_2. Since the symbols $a, b, \#$ are internal, the stack is not used by M_2 in its computation, and it only uses reversal-bounded counters. Thus we can remove the stack from M_2, and it becomes an NCM. From [4], M_2 can be converted to an equivalent NCM M_3 that runs in linear time. We get a contradiction, since it is easy to show (by a simple counting argument) that L_2 cannot be accepted by M_3. □

Every VPDA can be converted to an equivalent deterministic VPDA [1]. In contrast, from Theorem 9, we have:

Corollary 1. *There are VPCMs that cannot be converted to equivalent deterministic VPCMs.*

We can define a deterministic VPCM (DVPCM) in the obvious way: The machine has at most one choice of move at each step. In particular, if there is a transition on ε, there is no transition on any $a \in \Sigma$. We also assume that the machine always halts. Thus a DVPCM is a deterministic VPDA with reversal-bounded counters. (Note that a VPDA has no ε-transitions.) Since a VPDA can always be made deterministic, it follows that DVPCMs are strictly more powerful than deterministic VPDAs.

The first part of the next result is obvious. The second part follows from the first using Theorem 7.

Corollary 2

1. *The class of languages accepted by DVPCMs is closed under Boolean operations.*
2. *The containment and equivalence problems for DVPCMs are decidable.*

The second part of the above corollary does not hold for VPCMs, since the universe problem (does a given transducer accept all strings?) is already undecidable even for an NFA augmented with just one 1-reversal counter [4].

4 Applications

In this section we discuss some applications of the decidability of the emptiness and infinite problems for machines augmented with reversal-bounded counters.

4.1 Multiple Morphism Equivalence on Context-Free Languages

A morphism g is a mapping from $\Sigma^* \rightarrow \Delta^*$ such that $g(\varepsilon) = \varepsilon$, and $g(a_1 \cdots a_n) = g(a_1) \cdots g(a_n)$ for $n \geq 1, a_1, \ldots, a_n \in \Sigma$. The multiple morphism problem on CFLs is defined as follows: Given k pairs of morphisms $(g_1, h_1), \ldots, (g_k, h_k)$ and a CFL $L \subseteq \Sigma^*$, is it the case that for every $w \in L$, there exists an i such that $g_i(w) = h_i(w)$?

The following theorem was shown in [25]. We include a proof using the decidability of the emptiness problem for finite-crossing 3-phase 2NPCM.

Theorem 10. *The multiple morphism equivalence problem on CFLs is decidable.*

Proof. Let L be a language accepted by an NPDA M and $(g_1, h_1), \ldots, (g_k, h_k)$ be k pairs of morphisms. We construct a finite-crossing 3-phase 2NPCM M' with $2k$ counters, $C_1, \ldots, C_k, D_1, \ldots D_k$. M', when given input $w = a_1 \cdots a_n$, first makes $2k$ sweeps of the input without using the stack. On sweep $1 \leq i \leq k$, M' applies morphism g_i on w without recording the $g_i(w)$ but nondeterministically guessing some position p_i on the input and storing this position in counter C_i and the symbol a_{p_i} in the state. Then on the next k sweeps, M' applies morphism h_i on w and again nondeterministically guessing some position q_i on the input and storing this position in counter D_i and the symbol a_{q_i} in the state. Next, M checks that for each $1 \leq i \leq k$, the values of C_i and D_i are the same (this can be done by simultaneously decrementing the counters and confirm that they become zero at the same time), but that $a_{p_i} \neq a_{q_i}$. Finally, M' makes one last sweep of the input simulating the NPDA M on w (now using the stack) and accepts if M accepts. Clearly, $L(M') \neq \varnothing$ if and only if there is a $w \in L$ such that $g_i(w) \neq h_i(w)$ for all $1 \leq i \leq k$. The result follows, since the emptiness problem for finite-crossing 3-phase 2NPCMs is decidable (Theorem 4). □

Clearly, the above theorem still holds when the NPDA M is replaced by an NPCM (i.e., NPDA with reversal-bounded counters). Variations of the problem above can also be shown decidable. For example, it is decidable if $L' = \{w \mid w \in L$, there is no i such that $g_i(w) = h_i(w)\}$ is infinite, since the infiniteness problem for 3-phase 2NPCMs is decidable (Theorem 4).

4.2 Finite-Valuedness in Transducers

A transducer T is an acceptor with outputs. For example, an NPDT is a nondeterministic pushdown automaton with outputs. So the transitions are rules of the form:

$(q, a, Z) \to (p, x, y)$

where q, p are states, a is an input symbol or ε, Z is the top of the stack symbol, x is a (possibly null) string of stack symbols, and y is an output string (possibly ε). In this transition, T in state q, reads a, 'pops' Z and writes x on the stack, outputs string y, and enters state p.

We say that (u, v) is a transduction accepted by T if, when started in the initial state q_0, with input u, and the stack contains only the initial stack symbol Z_0, T enters an accepting state after reading u and producing v. The set of transductions accepted by T is denoted by $R(T)$. An NPCMT is an NPCM with outputs. Similarly, An NCMT, NFT, etc., is an NCM, NFA, etc. with outputs.

A transducer T is k-ambiguous ($k \geq 1$) if T with outputs ignored is a k-ambiguous acceptor. (Note that 1-ambiguous is the same as unambiguous). T is k-valued ($k \geq 1$) if for every u, there are at most k distinct strings v such that (u, v) is in $R(T)$. T is finite-valued if T is k-valued for some k.

The following result was recently shown in [29]. We include the proof in [29] to illustrate the reduction of the problem to the emptiness problem for NPCMs (which is decidable by Theorem 1).

Theorem 11. *The k-valuedness problem for 1-ambiguous NPCMT T is decidable.*

Proof. Consider the case $k = 1$, Given T with m reversal-bounded counters, we construct an NPCM M with two additional 1-reversal counters C_{12} and C_{21}.

Hence, M will have $m + 2$ reversal-bounded counters. M on input x, simulates T suppressing the outputs and accepts x if it finds two outputs y_1 and y_2 such that y_1 and y_2 disagree in some position p and x is accepted by T. In order to do this, during the simulation, M uses C_{12} and C_{21} to record the positions i and j (chosen nondeterministically) in y_1 and y_2, respectively, and the symbols a and b in these positions, such that $i = j$ and $a \neq b$. Clearly, a and b can be remembered in the state. Storing i and j need only increment C_{12} and C_{21} during the simulation. To check that $i = j$, after the simulation, M decrements C_{12} and C_{21} simultaneously and verifies that they reach zero at the same time. Note that since T is 1-ambiguous, M's accepting computation on x (except for the outputs) is unique and therefore the procedure just described can be accomplished by M on a single accepting run on the input. Clearly, T is 1-valued if and only if $L(M) = \varnothing$, which is decidable, since emptiness of NPCMs is decidable (by Theorem 1).

The above construction generalizes for any $k \geq 1$. Now M on input x, checks that there are at least $k + 1$ distinct outputs $y_1, \ldots, y_{k+1}$. M uses $k \times (k + 1)$ additional 1-reversal counters (so now M will have $m + k \times (k + 1)$ reversal-bounded counters.) In the simulation, for $1 \leq i \leq k+1$, M nondeterministically selects k positions $p_{i1}, \ldots, p_{i(i-1)}, p_{i(i+1)}, \ldots, p_{i(k+1)}$ in output y_i and records these positions in counters $C_{i1}, \ldots, C_{i(i-1)}, C_{i(i+1)}, \ldots, C_{i(k+1)}$ and the symbols at these positions in the state. At the end of the simulation, M accepts x if for all $1 \leq i, j \leq k + 1$ such that $i \neq j$, the symbol in position p_{ij} is different from the symbol in position p_{ji} and the value of counter C_{ij} is the same as the value of C_{ji}. □

The construction in the proof above does not work when T is k-ambiguous for any $k \geq 2$. This is because the computation of T on an input x may not be unique, so it is possible, e.g., that one accepting run on x produces output y_1 and a different accepting run on x produces output y_2. So to determine if $y_1 \neq y_2$, we need to simulate two runs on input x, i.e., M will no longer be one-way. In fact, the following was shown in [29]:

Theorem 12. *For any $k \geq 1$, it is undecidable, given a $(k + 1)$-ambiguous 1-reversal NPDT T (i.e., its stack makes only one reversal), whether T is k-valued.*

4.3 VPDTs with Reversal-Bounded Counters

Visibly pushdown transducers (VPDT) were introduced in [40], where ε-transitions that can produce outputs were allowed. Allowing such transitions makes some decision problems (e.g., single-valuedness) undecidable. Later, in [16], VPDTs that do not allow ε-transitions were investigated, where it was shown that the k-valuedness problem for VPDTs is decidable.

We can generalize the result above for VPDTs with reversal-bounded counters. A VPCMT T is a VPCM with outputs. Since a VPCM is allowed ε-transitions (where the stack is not used), we assume that on an ε-transition, the VPCMT can only output ε.

Theorem 13. *The k-valuedness problem for VPCMTs is decidable.*

4.4 Context-Free Transducers

A context-free transducer (CFT) T is a CFG with outputs, i.e., the rules are of the form $A \to (\alpha, y)$, where α is a string of terminals and nonterminals, and y is an output string (possibly ε). We assume that the underlying CFG G of T, i.e., the grammar obtained by deleting the outputs, has no ε-rules (i.e., rules of the form $A \to \varepsilon$) and unit-rules (i.e., rules of the form $A \to B$), where A, B are nonterminals). We also assume that all nonterminals are useful, i.e., reachable from the start nonterminal S and can reach a terminal string.

We consider only *leftmost* derivations in T, i.e., at each step, the leftmost nonterminal is expanded). Thus T generates transductions (u, v) (where u is a terminal string and v is an output string) derived in a sequence of rule applications in a *leftmost* derivation: $(S, \varepsilon) \Rightarrow^+ (u, v)$

A nonterminal A in the underlying CFG of a CFT is self-embedding if there is some leftmost derivation $A \Rightarrow^+ \alpha A \beta$ where α, β are strings of terminals and nonterminals. (Note that $|\alpha\beta| > 0$, since there are no ε-rules and unit-rules.)

The notions of ambiguous and finite-valuedness can also be defined for context-free grammars (CFGs) with outputs.

A CFT T is k-ambiguous for a given k (resp., finitely-ambiguous) if its underlying CFG is k-ambiguous (resp., finitely-ambiguous).

The next three lemmas were recently shown in [29]. We sketch the proofs given in [29] for completeness.

Lemma 1. *Let T be a finitely-ambiguous CFT with terminal and nonterminal alphabets Σ and N, respectively. Let G be its underlying finitely-ambiguous CFG. Let A be a nonterminal such that $A \Rightarrow^+ \alpha A \beta$, where $\alpha, \beta \in (\Sigma \cup N)^*$. Then this derivation (of $\alpha A \beta$) is unique.*

Proof. It is straightforward to show that if two distinct derivations $A \Rightarrow^+ \alpha A \beta$ exist, then A would not be finitely-ambiguous. $\square$

Lemma 2. *It is decidable, given a finitely-ambiguous CFT T, whether there exist a nonterminal A and a leftmost derivation $A \Rightarrow^+ \alpha A \beta$ for some $\alpha, \beta \in (\Sigma \cup N)^*$ (note that $\alpha\beta| > 0$), such that there are at least two distinct outputs generated in the derivation.*

Proof. Let A be a nonterminal and $L = \{w \mid w = \alpha A \beta$, for some $\alpha, \beta \in (\Sigma \cup N)^*$ such that $|\alpha\beta| > 0$, and $A \Rightarrow^+ \alpha A \beta$ produces at least two distinct outputs$\}$. (Thus $L \subseteq (\Sigma \cup N)^*$.)

We construct an NPCM M to accept L. M, when given input w, tries to simulate a leftmost derivation $A \Rightarrow^+ \alpha A \beta$ (which, if it exists, is unique by Lemma 1) and checks that there are at least two distinct outputs generated in the derivation. Initially, A is the only symbol on the stack. Each derivation step is of the form $B \to x\varphi$, where x is in Σ^* and φ is in $N(\Sigma \cup N)^* \cup \{\varepsilon\}$. If B is the top of the stack, M simulates the step by popping B, checking that the remaining input segment to be read has prefix x (if $x \neq \varepsilon$), and pushing φ on the pushdown stack (if $\varphi \neq \varepsilon$). It uses two 1-reversal counters C_1 and C_2 to check that there is a discrepancy in the outputs corresponding to the derivation $A \Rightarrow^+ \alpha A \beta$. Since the derivation $A \Rightarrow^+ \alpha A \beta$ is unique, this can be done in the same manner as described in the proof of Theorem 11.

At some point in the derivation, M guesses that the stack contains a string of the form $z = \gamma_1 A \gamma_2$, where $\gamma_1, \gamma_2 \in (\Sigma \cup N)^*$. M then pops the stack and checks that the remaining input yet to be read is $\gamma_1 A \gamma_2$ and accepts if there were two distinct outputs generated in the derivation.

It is easily verified that $L(M) = L$. The result follows since the emptiness problem for NPCMs is decidable by Theorem 1. □

Lemma 3. *Let T be a finitely-ambiguous CFT and G be its underlying CFG. Suppose for some nonterminal A, there is a leftmost derivation $A \Rightarrow^+ \alpha A \beta$ for some $\alpha, \beta \in (\Sigma \cup N)^*$, with $|\alpha\beta| > 0$ such that there are at least two distinct outputs generated in the derivation. Then T is not finite-valued.*

Proof. Suppose there is a self-embedding nonterminal A and a derivation $A \Rightarrow^+ \alpha A \beta$, where $|\alpha\beta| > 0$ and there are at least two distinct strings y_1 and y_2 that are outputted in the derivation. Since all nonterminals are useful, we have in G, $S \Rightarrow^* wAx \Rightarrow^+ w\alpha A \beta x \Rightarrow^* w\alpha^k A \beta^k x \Rightarrow^+ wu^k z v^k x$ for some terminal strings w, u, z, v, x, for all $k \geq 1$. Let y_0, y_3, y_4, y_5 be the outputs in the derivations $S \Rightarrow^* wAx$, $\alpha \Rightarrow^* u$, $A \Rightarrow^+ z$, and $\beta \Rightarrow^* v$, respectively. There are two cases: *Case 1.* $|y_1| = |y_2|$ but $y_1 \neq y_2$, and Case 2. $|y_1| \neq |y_2|$. In both cases, the CFT T can be shown to be finite-valued. □

4.4.1 Linear Context-Free Transducers

A linear context-free transducer (LCFT) is a CFT whose underlying grammar is a linear CFG (LCFG). Thus, the rules are of the form $A \to (uBv, y)$ or $A \to (u, y)$, where A, B are nonterminals, u, v are terminal strings with $|uv| > 0$, and y is an output string. The following converse of Lemma 3 was shown in [29]:

Lemma 4. *Let T be a finitely-ambiguous LCFT and G be its underlying LCFG. Suppose there is no nonterminal A for which there is a leftmost derivation $A \Rightarrow^+ uAv$ for some $u, v \in \Sigma^*$, with $|uv| > 0$ such that there are at least two distinct outputs generated in the derivation. Then T is finite-valued.*

From Lemmas 2, 3, and 4:

Theorem 14. *It is decidable, given a finitely-ambiguous LCFT T, whether it is finite-valued.*

When the LCFT is not finitely-ambiguous, Theorem 14 does not hold:

Theorem 15. *It is undecidable, given a LCFT T, whether it is finite-valued.*

Proof. In [44], it was shown that there is a class of LCFGs for which every grammar in the class is either unambiguous or unboundedly ambiguous, but determining which is the case is undecidable. Let G be a LCFG in this class. Number the rules in G and construct a LCFT T which outputs the rule number corresponding to each rule. The result then follows. □

Lemmas 1, 2, and 3 were proved for finitely-ambiguous CFTs. However, Lemma 4 does not hold for finitely-ambiguous CFTs. For consider the 1-ambiguous CFT: $S \to (SA, 0) \mid (a, 0)$, $A \to (a, \{0, 1\})$, where S, A are nonterminals, a is a terminal symbol, and $0, 1$ are output symbols. This CFT satisfies the hypothesis of Lemma 4, but it is not finite-valued.

4.4.2 Left-Derivation-Bounded Context-Free Transducers

A CFG is left-derivation-bounded (LDBCFT) if there is an $s \geq 1$ such that for any nonterminal A, every sentential form derivable from A when restricted to *leftmost* derivation has at most s nonterminals [42]. If the derivation is *not restricted* to leftmost derivation, the grammar is called derivation-bounded [17]. If the upper bound s must hold for *all* derivations of the sentential form, the CFG is called non-terminal bounded. Clearly, every nonterminal-bounded CFG is left-derivation-bounded, which, in turn, is derivation-bounded. It was shown in [42] that the language classes defined by these grammars form a strict hierarchy.

The following was proved in [29]:

Theorem 16. *It is decidable, given a finitely-ambiguous LDBCFT T, whether it is finite-valued.*

4.4.3 Context-Free Transducers

Let T be a CFT and G be its underlying CFG. For $A \in N$ and $\alpha \in (\Sigma \cup N)^*$, let $NT(A, \alpha)$ be the number of A's in α. Let $\#$ be a symbol not in the terminal alphabet of G. Let $L_A = \{\alpha\#^d \mid S \Rightarrow^+ \alpha, NT(A, \alpha) \geq d\}$. Clearly, G is left-derivation-bounded if and only if there is a nonterminal A such that L_A is infinite. Call such an A an abundant nonterminal.

The proofs of Lemma 5 and Corollary 3 below use the decidability of the infiniteness problem for NPCMs (Theorem 1).

Lemma 5. *It is decidable, given a CFG G and a nonterminal A, whether A is abundant. Hence, it is decidable whether G is left-derivation-bounded.*

The above lemma generalizes to:

Corollary 3. *It is decidable, given a CFG G and distinct nonterminals $A_1, \ldots,$ A_k, whether $L_{(A_1,\ldots,A_k)} = \{\alpha\#^d \mid S \Rightarrow^+ \alpha, NT(A_1, \alpha) \geq d, \ldots, NT(A_k, \alpha) \geq d\}$ is infinite.*

We now assume that the CFT is 1-ambiguous.

Theorem 17. *It is decidable, given a 1-ambiguous CFT T , whether it is finite-valued.*

Proof. (Sketch) Let G be the underlying (1-ambiguous) CFG of T. If G is left-derivation-bounded (which is decidable by Lemma 5), then we can decide if T is finite-valued (by Theorem 16).

If G is not left-derivation-bounded, let A be an abundant nonterminal. Determine if there is a string of terminals and nonterminals β such that $A \Rightarrow^+ \beta$ and in this derivation, there are at least two distinct output strings. Call such a nonterminal multi-valued. Since G is 1-ambiguous, this property can be decided by a construction similar to those in the proofs of Theorem 11 and Lemma 2. [NOTE: This is the part where we need T (hence G) to be 1-ambiguous, since this is unlike a self-embedding derivation for which Lemma 2 applies, where the derivation is unique if T is finitely ambiguous.]

Clearly, if G has an abundant multi-valued nonterminal, then T is not finite-valued. Now suppose G does not have any abundant multi-valued nonterminal. We consider two cases:
Case 1. G has a self-embedding nonterminal B such that $B \Rightarrow^+ \gamma_1 B \gamma_2$ for some γ_1, γ_2 and in this derivation, there are at least two distinct outputs. Then T is not finite-valued (by Lemma 3).
Case 2. Assume G does not have a self-embedding nonterminal as described in case 1. We can show that T is finite-valued. We omit the proof.
From (1) and 2(a) and 2(b), it follows that T is finite-valued. □

4.5 Containment and Equivalence Problems for Transducers

It is known that the equivalence problem for two-way deterministic finite transducers (2DFTs) is decidable [21]. However, if nondeterminism is allowed, the problem becomes undecidable even for one-way nondeterministic finite transducers (NFTs) [19]. The undecidability holds even for NFTs operating on a unary input (or output) alphabet [28]. For single-valued (i.e., 1-valued) NFTs, the problem becomes decidable [23], and the decidability result was later extended to finite-valued NFTs in [9]. The complexity of the problem was subsequently derived in [43].

In [38], we generalized the result of [21] for some models of two-way transducers (with input end markers # and \$) augmented with reversal-bounded counters. Call the nondeterministic (resp., deterministic) version 2NCMT (resp., 2DCMT). The relation defined by such a transducer T is $R(T) = \{(x, y) \mid T$, when started in its initial state on the left end marker of $\#x\$$, outputs y and falls off the right end marker in an accepting state$\}$. The transducer is finite-crossing if there is some fixed k such that in every accepting computation on any input $\#x\$$, the number of times the input head crosses the boundary between any two adjacent symbols of $\#x\$$ is at most k. Note that the number of turns (i.e., changes in direction from left-to-right and right-to-left and vice-versa) the input head makes on the input may be unbounded. Also note that the requirement is only for accepting computations. So if $R(T) = \varnothing$, then T is finite-crossing and, in fact, k-crossing for any k. We assume that when we are given a finite-crossing machine, the integer k for which the machine is k-crossing is also specified.

Clearly, every 2DFT is finite-crossing because, by definition, a valid computation must always fall off the right end marker in an accepting state. So no input cell can be visited twice in the same state; otherwise, the machine will never fall off the right end marker. Hence, a 2DFT is a special case of finite-crossing 2DCMT. But the latter is more powerful. Consider the relation $R = \{(xd^k, y) \mid x \in \Sigma^+, k > 0,\ |x| \geq 2k,\ x = x_1x_2x_3,\ |x_1| = |x_3| = k,\ y = x_3x_2x_1\}$, where d is a symbol not in alphabet Σ. R can be implemented on a finite-crossing 2DCMT using two reversal-bounded counters (in fact, the head need only make finite-turn on the input tape), but cannot be implemented on a 2DFT. On the other hand, for 2NFT, we can no longer say it is finite-crossing since the machine can always decide to fall off the right end marker, even though a cell has been visited more than once in the same state; e.g., $\{(x, x^n) | x \in \Sigma^+, n > 0\}$ can be implemented on a 2NFT. However, a finite-crossing 2NCMT is more powerful than a finite-crossing 2NFT, as R cannot also be implemented on a finite-crossing 2NFT.

It was shown in [38] that the following problems are decidable:

1. Given a finite-crossing 2NCMT T_1 and a finite-crossing 2DCMT T_2, is $R(T_1) \subseteq R(T_2)$? Hence, equivalence of finite-crossing 2DCMTs is decidable.
2. Given a one-way nondeterministic pushdown transducer with reversal-bounded counters (NPCMT) T_1 and a finite-crossing 2DCMT T_2, is $R(T_1) \subseteq R(T_2)$?
3. Given a finite-crossing 2NCMT T_1 and a DPCMT T_2 (the deterministic version of NPCMT), is $R(T_1) \subseteq R(T_2)$?
4. Given a finite-crossing 2DCMT T_1 and a DPCMT T_2, is $R(T_1) = R(T_2)$?

In the above results, the "finite-crossing" assumption is necessary, since it can be shown that when the two-way input is unrestricted, the equivalence problem becomes undecidable. Also, the NPCMT and DPCMT in (2), (3) and (4) above cannot be generalized to be two-way, since it can be shown that it is undecidable to determine, given a 2DPCMT T, whether $R(T) = \varnothing$, even when T makes only two turns on the input. However, the following was proved in [38]:

5. It is decidable to determine, given two finite-crossing 2DPCMTs whose inputs come from a bounded language (i.e., from $w_1^* \cdots w_k^*$ for some non-null strings $w_1, \ldots, w_k$) T_1 and T_2, whether $R(T_1) \subseteq R(T_2)$. (Hence, equivalence is also decidable.)

The proofs for the results above use the decidability of the emptiness problems in Section 2, in particular, the decidability of emptiness for 3-phase finite-crossing 2NPCMs.

The containment and equivalence between single-valued finite-crossing two-way nondeterministic finite transducers (2NFTs) and various finite-crossing two-way transducers with reversal-bounded counters were investigated in [38], where the following problems were shown to be decidable:

6. Given a finite-crossing 2NCMT (or an NPCMT) T_1 and a single-valued finite-crossing 2NFT T_2, is $R(T_1) \subseteq R(T_2)$?

7. Given a single-valued finite-crossing 2NFT T_1 and a finite-crossing 2DCMT (or a DPCMT) T_2, is $R(T_1) \subseteq R(T_2)$?
8. Given a single-valued finite-crossing 2NFT T_1 and a finite-crossing 2DCMT (or a DPCMT) T_2, is $R(T_1) = R(T_2)$?

In [2,3], deterministic and nondeterministic versions of streaming string transducers (SSTs, for short) were investigated. An SST uses a finite set of variables ranging over strings from the output alphabet in the process of making a single pass through the input string to produce the output string. It was shown in [2] that deterministic SSTs and two-way deterministic finite transducers (2DFTs) are equally expressive. For nondeterministic SSTs, they are incomparable with 2-way nondeterministic finite transducers (2NFTs) in terms of the expressiveness, although they are more expressive than 1-way nondeterministic finite transducers (NFTs).

As "variables" used in SSTs can be regarded as a form of auxiliary memory, it would be of interest to further investigate the relationship between two-way (finite-crossing) transducers augmented with auxiliary memory and SSTs with respect to the expressiveness as well as various decision problems. As was pointed out earlier, the relation $T = \{(xd^k, y) \mid x \in \Sigma^+, k > 0,\ |x| \geq 2k,\ x = x_1x_2x_3,\ |x_1| = |x_3| = k,\ y = x_3x_2x_1\}$, where d is a symbol not in alphabet Σ can be implemented on a finite-crossing 2DCMT, but cannot be implemented on a 2DFT. Hence, finite-crossing 2DCMTs are more expressive than deterministic SSTs, as the latter are equivalent to 2DFTs. The relationship between nondeterministic SSTs and various 2-way (finite-crossing) transducers with auxiliary memory remains unknown.

Finally, we consider the containment and equivalence problems for VPCMTs and DVPCMTs. As we mentioned earlier, the containment and equivalence problems for NFTs is undecidable. Hence these problems are also undecidable for VPCMTs. However, we can prove:

Theorem 18. *The following problems are decidable:*

1. *Given a VPCMT T_1 and a DVPCMT T_2, is $R(T_1) \subseteq R(T_2)$?*
2. *Given two DVPCMTs T_1 and T_2, is $R(T_1) = R(T_2)$?*

Proof. Clearly, we only need to prove (1). Let M_1 be the underlying VPCM of T_1 and M_2 be the underlying DVPCM of T_2. Thus, $L(M_1) = \mathit{domain}(R(T_1))$ and $L(M_2) = \mathit{domain}(R(T_2))$.

First we determine if $L(M_1) \subseteq L(M_2)$. This is decidable, since from Theorem 2, we can construct a DVPCM M_3 to accept the complement of $L(M_2)$. Then from Theorem 8 we can construct a VPCM M_4 to accept $L(M_1) \cap L(M_3)$. Clearly $L(M_1) \subseteq L(M_2)$ if and only if $L(M_4) = \varnothing$, which is decidable by Theorem 7.

Obviously, if $L(M_1) \nsubseteq L(M_2)$, then $R(T_1) \nsubseteq R(T_2)$. Otherwise, $R(T_1) \nsubseteq R(T_2)$ if and only if there exists an x such that the following two conditions are satisfied:

(a) For some y, (x, y) is in $R(T_1)$, and

(b) x is in $domain(R(T_2))$ and the only z such that (x, z) is in $R(T_2)$ is different from y.

Given T_1, M_1, T_2, M_2, we construct a VPCM M such that $L(M) \neq \varnothing$ if and only if the conditions above are satisfied (we omit the construction here). The result follows since we can decide if $L(M) = \varnothing$ by Theorem 7. $\square$

References

1. Alur, R., Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Proc. of STOC 2004, pp. 202–211 (2004)
2. Alur, R., Cerny, P.: Expressiveness of streaming string transducers. In: Proc. 30th Annual Conf. on Foundations of Software Technology and Theoretical Computer Science, pp. 1–12 (2010)
3. Alur, R., Deshmukh, J.: Nondeterministic streaming string transducers. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 1–20. Springer, Heidelberg (2011)
4. Baker, B., Book, R.: Reversal-bounded multipushdown machines. J. Comput. System Sci. 8, 315–332 (1974)
5. Bouchy, F., Finkel, A., San Pietro, P.: Dense-choice counter machines revisited. In: Proc. of INFINITY 2009, pp. 3–22 (2009)
6. Bouchy, F., Finkel, A., Sangnier, A.: Reachability in timed counter systems. Electr. Notes Theor. Comput. Sci. 239, 167–178 (2009)
7. Cadilhac, M., Finkel, A., McKenzie, P.: Affine Parikh automata. RAIRO - Theor. Inf. and Applic. 46(4), 511–545 (2012)
8. Cadilhac, M., Finkel, A., McKenzie, P.: Unambiguous constrained Automata. Int. J. Found. Comput. Sci. 24(7), 1099–1116 (2013)
9. Culik, K., Karhumaki, J.: The equivalence of finite valued transducers (on HDTOL languages) is decidable. Theoret. Comput. Sci. 47, 71–84 (1986)
10. Dang, Z.: Binary reachability analysis of pushdown timed automata with dense clocks. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 506–517. Springer, Heidelberg (2001)
11. Dang, Z., Bultan, T., Ibarra, O.H., Kemmerer, R.A.: Past pushdown timed automata. In: Watson, B.W., Wood, D. (eds.) CIAA 2001. LNCS, vol. 2494, pp. 74–86. Springer, Heidelberg (2003)
12. Dang, Z., Ibarra, O.H., Bultan, T., Kemmerer, R.A., Su, J.: Binary reachability analysis of discrete pushdown timed automata. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 69–84. Springer, Heidelberg (2000)
13. Dang, Z., Ibarra, O.H., Miltersen, P.B.: Liveness verification of reversal-bounded multicounter machines with a free counter. In: Hariharan, R., Mukund, M., Vinay, V. (eds.) FSTTCS 2001. LNCS, vol. 2245, pp. 132–143. Springer, Heidelberg (2001)
14. Dang, Z., Ibarra, O.H., Sun, Z.-W.: On two-way nondeterministic finite automata with one reversal-bounded counter. Theor. Comput. Sci. 330(1), 59–79 (2005)
15. Finkel, A., Sangnier, A.: Reversal-bounded counter machines revisited. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 323–334. Springer, Heidelberg (2008)
16. Filiot, E., Raskin, J.-F., Reynier, P.-A., Servais, F., Talbot, J.-M.: Properties of visibly pushdown transducers. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 355–367. Springer, Heidelberg (2010)

17. Ginsburg, S., Spanier, E.H.: Derivation-bounded languages. J. Comput. System Sci. 2(3), 228–250 (1968)
18. Ginsburg, S., Spanier, E.H.: Bounded ALGOL-like languages. T. Am. Math. Soc. 113, 333–368 (1964)
19. Griffiths, T.: The unsolvability of the equivalence problem for Λ-free nondeterministic generalized sequential machines. J. Assoc. Comput. Mach. 15, 409–413 (1968)
20. Gurari, E., Ibarra, O.H.: The complexity of decision problems for finite-turn multicounter machines. J. Comput. Syst. Sci. 22, 220–229 (1981)
21. Gurari, E.: The equivalence problem for deterministic two-way sequential transducers is decidable. SIAM J. Comput. 11, 448–452 (1982)
22. Gurari, E., Ibarra, O.H.: Two-way counter machines and Diophantine equations. J. ACM 29(3), 863–873 (1982)
23. Gurari, E., Ibarra, O.H.: A note on finite-valued and finitely ambiguous transducers. Math. Systems Theory 16, 61–66 (1983)
24. Hague, M., Lin, A.W.: Model checking recursive programs with numeric data types. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 743–759. Springer, Heidelberg (2011)
25. Harju, T., Ibarra, O.H., Karhumaski, J., Salomaa, A.: Some decision problems concerning semilinearity and commutation. J. Comput. Syst. Sci. 65(2), 278–294 (2002)
26. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata, Languages and Computation. Addison-Wesley (1978)
27. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. of the ACM 25(1), 116–133 (1978)
28. Ibarra, O.H.: The unsolvability of the equivalence problem for ε-free NGSM's with unary input (output) alphabet and applications. SIAM J. Computing 7, 524–532 (1978)
29. Ibarra, O.H.: On the ambiguity, finite-valuedness, and lossiness problems in acceptors and transducers. In: Proc. of CIAA 2014 (to appear, 2014)
30. Ibarra, O.H.: On decidability and closure properties of language classes with respect to bio-operations. In: Proc. of DNA 20 (to appear)
31. Ibarra, O.H., Bultan, T., Su, J.: Reachability analysis for some models of infinite-state transition systems. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 183–198. Springer, Heidelberg (2000)
32. Ibarra, O.H., Dang, Z.: On the solvability of a class of diophantine equations and applications. Theor. Comput. Sci. 352(1-3), 342–346 (2006)
33. Ibarra, O.H., Dang, Z., Egecioglu, O., Saxena, G.: Characterizations of Catalytic Membrane Computing Systems. In: Rovan, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 480–489. Springer, Heidelberg (2003)
34. Ibarra, O.H., Jiang, T., Tran, N., Wang, H.: New decidability results concerning two-way counter machines. SIAM J. Computing (24), 123–137 (1995)
35. Ibarra, O.H., Seki, S.: Characterizations of bounded semilinear languages by one-way and two-way deterministic machines. In: Proc. 13th Int. Conf. on Automata and Formal Languages, pp. 211–224 (2011)
36. Ibarra, O.H., Seki, S.: Semilinear sets and counter machines: a brief survey (submitted)
37. Ibarra, O.H., Su, J., Dang, Z., Bultan, T., Kemmerer, R.A.: Counter machines: decidable properties and applications to verification problems. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, pp. 426–435. Springer, Heidelberg (2000)

38. Ibarra, O.H., Yen, H.-C.: On the containment and equivalence problems for two-way transducers. Theor. Comput. Sci. 429, 155–163 (2012)
39. Minsky, M.: Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines. Ann. of Math. (74), 437–455 (1961)
40. Raskin, J.-F., Servais, F.: Visibly pushdown transducers. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 386–397. Springer, Heidelberg (2008)
41. Verma, K.N., Seidl, H., Schwentick, T.: On the complexity of equational horn clauses. In: Nieuwenhuis, R. (ed.) CADE 2005. LNCS (LNAI), vol. 3632, pp. 337–352. Springer, Heidelberg (2005)
42. Walljasper, S.J.: Left-derivation bounded languages. J. Comput. and System Sci. 8(1), 1–7 (1974)
43. Weber, A.: Decomposing finite-valued transducers and deciding their equivalence. SIAM. J. on Computing 22, 175–202 (1993)
44. Wich, K.: Exponential ambiguity of context-free grammars. In: Proc. of 4th Int. Conf. on Developments in Language Theory, pp. 125–138. World Scientific (1999)
45. Xie, G., Dang, Z., Ibarra, O.H.: A Solvable class of quadratic diophantine equations with applications to verification of infinite-state systems. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 668–680. Springer, Heidelberg (2003)
46. Xie, G., Dang, Z., Ibarra, O.H., Miltersen, P.B.: Dense counter machines and verification problems. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 93–105. Springer, Heidelberg (2003)