# Insertion Operations on Deterministic Reversal-Bounded Counter Machines

Joey Eremondi[1], Oscar H. Ibarra[2], and Ian McQuillan[3](✉)

[1] Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
j.s.eremondi@students.uu.nl

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

[3] Department of Computer Science, University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada
mcquillan@cs.usask.ca

**Abstract.** Several insertion operations are studied applied to languages accepted by one-way and two-way deterministic reversal-bounded multicounter machines. These operations are defined by the ideals obtained from relations such as the prefix, infix, suffix and outfix relations. The insertion of regular languages and other languages into deterministic reversal-bounded multicounter languages is also studied. The question of whether the resulting languages can always be accepted by deterministic machines with the same number of turns on the input tape, the same number of counters, and reversals on the counters is investigated. In addition, the question of whether they can always be accepted by increasing either the number of input tape turns, counters, or counter reversals is addressed. The results in this paper form a complete characterization based on these parameters. Towards these new results, we use a technique for simultaneously showing a language cannot be accepted by both one-way deterministic reversal-bounded multicounter machines, and by two-way deterministic machines with one reversal-bounded counter.

**Keywords:** Automata and logic · Counter machines · Insertion operations · Reversal-bounds · Determinism · Finite automata

## 1 Introduction

One-way deterministic multicounter machines are deterministic finite automata augmented by a fixed number of counters, which can each be independently increased, decreased or tested for zero. If there is a bound on the number of switches each counter makes between increasing and decreasing, then the

The research of O. H. Ibarra was supported, in part, by NSF Grant CCF-1117708. The research of I. McQuillan was supported, in part, by the Natural Sciences and Engineering Research Council of Canada.


A.-H. Dediu et al. (Eds.): LATA 2015, LNCS 8977, pp. 200–211, 2015.
DOI: 10.1007/978-3-319-15579-1_15

machine is reversal-bounded [1,8]. The family of languages accepted by one-way deterministic reversal-bounded multicounter machines (denoted by DCM) is interesting as it is more general than regular languages, but still has a decidable emptiness, infiniteness, equivalence, inclusion, universe and disjointness problems [8]. Moreover, these problems remain decidable if the machines operate with two-way input that is finite-crossing in the sense that there is a fixed $k$ such that the number of times the boundary between any two adjacent input cells is crossed is at most $k$ times [4].

Reversal-bounded counter machines (both deterministic and nondeterministic) have been extensively studied. Many generalizations have been investigated, and they have found applications in areas such as verification of infinite-state systems, membrane computing systems, Diophantine equations, etc.

In this paper, we study various insertion operations on deterministic reversal-bounded multicounter languages. Common word and language relations are the prefix, suffix, infix and outfix relations. For example, $w$ is an infix of $z$, written $w \leq_{\rm i} z$, if $z = xwy$, for some $x, y \in \Sigma^*$. Viewed as an operation on the first component of the relation, $\leq_{\rm i} (w) = \{z \mid w \leq_{\rm i} z, z \in \Sigma^*\}$, which is equal to the set of all words with $w$ as infix, which is $\Sigma^* w \Sigma^*$. If we consider the inverse of this relation, $z \leq_{\rm i}^{-1} w$, if $z = xwy$, then viewing this as an operation, $\leq_{\rm i}^{-1}$ $(z) = \{w \mid z \leq_{\rm i}^{-1} w, w \in \Sigma^*\} = \{w \mid w \leq_{\rm i} z\}$, the set of all infixes of $z$. These can be extended to operations on languages. The prefix, suffix, infix and outfix operations can be defined on languages in this way, along with their inverses. This is the approach taken in [10]. Using the more common notation of $\inf(L)$ for the set of infixes of $L$, then $\inf^{-1}(L) = \Sigma^* L \Sigma^*$, the set of all words having a word in $L$ as an infix. This is the same as what is often called the *two-sided ideal*, or the *infix ideal* [10]. For the suffix operation, $\mathrm{suff}(L) = (\Sigma^*)^{-1} L$, and $\mathrm{suff}^{-1}(L) = \Sigma^* L$, with the latter being called the *left ideal*, or the *suffix ideal*. For prefix, $\mathrm{pref}(L) = L(\Sigma^*)^{-1}$, and $\mathrm{pref}^{-1}(L) = L\Sigma^*$, the *prefix ideal*, or the *right ideal*. The inverse of each operation defines a natural insertion operation.

We will examine the insertion operations defined by the inverse of the prefix, suffix, infix, outfix and embedding relations, and their effects on deterministic reversal-bounded multicounter languages. We will also examine certain standard generalizations of these operations such as left and right concatenation with regular or more general languages. In particular, if we start with a language that can be accepted with a parameterized number of input tape turns, counters, and reversals on the counters, is the result of the various insertion operations always accepted with the same type of machines? And if not, can they always be accepted by increasing either the turns on the input tape, counters, or reversals on the counters? Results in this paper form a complete characterization in this regard, and are summarized in Section 5. Surprisingly, even if we have languages accepted by deterministic 1-reversal bounded machines with either one-way input and 2 counters, or 1 counter and 1 turn on the input, then concatenating $\Sigma^*$ to the right can result in languages that can neither be accepted by DCM machines (any number of reversal-bounded counters), nor by two-way deterministic reversal-bounded 1-counter machines (2DCM(1), which have no bound on input turns). This is in contrast to deterministic pushdown languages which are closed under right concatenation with regular languages [6]. In addition, concatenating $\Sigma^*$ to the left of a one-way 1-reversal-bounded one counter machine can create languages that are neither in DCM nor 2DCM(1). Furthermore, as a consequence of the results in this paper, it is evident that the right input end-marker strictly increases the power for even one-way deterministic reversal-bounded multicounter languages when there are at least two counters. This is usually not the case for various classes of one-way machines. To do this, a new mode of acceptance, by *final state without end-marker*, is defined and studied.

Most non-closure results in this paper use a technique that simultaneously shows languages are not in DCM and not in DCM(1). The technique does not rely on any pumping arguments. A similar technique was used in [2] for showing that there is a language accepted by a deterministic pushdown automaton whose stack makes only one reversal (1-reversal DPDA) that cannot be accepted by any one-way nondeterministic reversal-bounded multicounter machine (NCM).

## 2 Preliminaries

The set of non-negative integers is represented by $\mathbb{N}_0$, and positive integers by $\mathbb{N}$. For $c \in \mathbb{N}_0$, let $\pi(c)$ be 0 if $c = 0$, and 1 otherwise.

We use standard notations for formal languages, referring the reader to [6,7]. The empty word is denoted by $\lambda$. We use $\Sigma$ and $\Gamma$ to represent finite alphabets, with $\Sigma^*$ as the set of all words over $\Sigma$ and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. For a word $w \in \Sigma^*$, if $w = a_1 \cdots a_n$ where $a_i \in \Sigma$, $1 \le i \le n$, the length of $w$ is denoted by $|w| = n$, and the reversal of $w$ is denoted by $w^R = a_n \cdots a_1$. The number of $a$'s, for $a \in \Sigma$, in $w$ is $|w|_a$. Given a language $L \subseteq \Sigma^*$, the complement of $L$, $\Sigma^* \setminus L$ is denoted by $\overline{L}$.

**Definition 1.** *For a language $L \subseteq \Sigma^*$, we define the prefix, inverse prefix, suffix, inverse suffix, infix, inverse infix, outfix and inverse outfix operations, respectively:*

| | |
|---|---|
| $\mathrm{pref}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$ | $\mathrm{pref}^{-1}(L) = \{wx \mid w \in L, x \in \Sigma^*\}$ |
| $\mathrm{suff}(L) = \{w \mid xw \in L, x \in \Sigma^*\}$ | $\mathrm{suff}^{-1}(L) = \{xw \mid w \in L, x \in \Sigma^*\}$ |
| $\mathrm{inf}(L) = \{w \mid xwy \in L, x, y \in \Sigma^*\}$ | $\mathrm{inf}^{-1}(L) = \{xwy \mid w \in L, x, y \in \Sigma^*\}$ |
| $\mathrm{outf}(L) = \{xy \mid xwy \in L, w \in \Sigma^*\}$ | $\mathrm{outf}^{-1}(L) = \{xwy \mid xy \in L, w \in \Sigma^*\}$ |

We generalize the outfix relation to the notion of embedding [10]:

**Definition 2.** *The $m$-embedding of a language $L \subseteq \Sigma^*$ is the following set:* $\mathrm{emb}(L, m) = \{w_0 \cdots w_m \mid w_0 x_1 \cdots w_{m-1} x_m w_m \in L,\ w_i \in \Sigma^*, 0 \le i \le m, x_j \in \Sigma^*, 1 \le j \le m\}$.

*We define the inverse as follows:* $\mathrm{emb}^{-1}(L, m) = \{w_0 x_1 \cdots w_{m-1} x_m w_m \mid w_0 \cdots w_m \in L,\ w_i \in \Sigma^*, 0 \le i \le m, x_j \in \Sigma^*, 1 \le j \le m\ \}$

Note that $\mathrm{outf}(L) = \mathrm{emb}(L, 1)$ and $\mathrm{outf}^{-1}(L) = \mathrm{emb}^{-1}(L, 1)$.

A language $L$ is called *prefix-free* if, for all words $x, y \in L$, where $x$ is a prefix of $y$, then $x = y$.

A *one-way k-counter machine* is a tuple $M = (k, Q, \Sigma, \$, \delta, q_0, F)$, where $Q, \Sigma, \$, q_0, F$ are respectively the finite set of states, the input alphabet, the right end-marker, the initial state in $Q$, and the set of final states, which is a subset of $Q$. The transition function $\delta$ (defined as in [8] except with only a right end-marker since these machines only use one-way inputs) is a mapping from $Q \times (\Sigma \cup \{\$\}) \times \{0,1\}^k$ into $Q \times \{\mathrm{S}, \mathrm{R}\} \times \{-1, 0, +1\}^k$, such that if $\delta(q, a, c_1, \ldots, c_k)$ contains $(p, d, d_1, \ldots, d_k)$ and $c_i = 0$ for some $i$, then $d_i \geq 0$ to prevent negative values in any counter. The symbols S are R indicate the direction of input tape head movement, either *stay* or *right* respectively. The machine $M$ is *deterministic* if $\delta$ is a function. The machine $M$ is *non-exiting* if there are no transitions defined on final states. A *configuration* of $M$ is a $k+2$-tuple $(q, w\$, c_1, \ldots, c_k)$ representing the fact that $M$ is in state $q$, with $w \in \Sigma^*$ still to read as input, and $c_1, \ldots, c_k \in \mathbb{N}_0$ are the contents of the $k$ counters. The derivation relation $\vdash_M$ is defined between configurations, where $(q, aw, c_1, \ldots, c_k) \vdash_M (p, w', c_1+d_1, \ldots, c_k+d_k)$, if $(p, d, d_1, \ldots, d_k) \in \delta(q, a, \pi(c_1), \ldots, \pi(c_k))$ where $d \in \{\mathrm{S}, \mathrm{R}\}$ and $w' = aw$ if $d = \mathrm{S}$, and $w' = w$ if $d = \mathrm{R}$. We let $\vdash_M^*$ be the reflexive, transitive closure of $\vdash_M$. And, for $m \in \mathbb{N}_0$, let $\vdash_M^m$ be the application of $\vdash_M$ $m$ times. A word $w \in \Sigma^*$ is accepted by $M$ if $(q_0, w\$, 0, \ldots, 0) \vdash_M^* (q, \$, c_1, \ldots, c_k)$, for some $q \in F$, and $c_1, \ldots, c_k \in \mathbb{N}_0$. The language accepted by $M$, denoted by $L(M)$, is the set of all words accepted by $M$.

The machine $M$ is $l$-reversal bounded if, in every accepting computation, the count on each counter alternates between increasing and decreasing at most $l$ times. We will sometimes refer to a multicounter machine as being in $\mathsf{DCM}(k, l)$, if it has $k$ $l$-reversal bounded counters.

We denote by $\mathsf{NCM}(k, l)$ the family of languages accepted by one-way nondeterministic $l$-reversal-bounded $k$-counter machines. We denote by $\mathsf{DCM}(k, l)$ the family of languages accepted by one-way deterministic $l$-reversal-bounded $k$-counter machines. The union of the families of languages are denoted by $\mathsf{NCM} = \bigcup_{k,l \geq 0} \mathsf{DCM}(k, l)$ and $\mathsf{DCM} = \bigcup_{k,l \geq 0} \mathsf{DCM}(k, l)$.

Given a $\mathsf{DCM}$ machine $M = (k, Q, \Sigma, \$, \delta, q_0, F)$, the language accepted by *final state without end-marker* is the set of words $w$ such that $(q_0, w\$, 0, \ldots, 0) \vdash_M^* (q', a\$, c_1', \ldots, c_k') \vdash_M (q, \$, c_1, \ldots, c_k)$, for some $q \in F$, $q' \in Q$, $a \in \Sigma$, $c_i, c_i' \in \mathbb{N}_0, 1 \leq i \leq k$. Such a machine does not "know" when it has reached the end-marker \$. The state that the machine is in when the last letter of input from $\Sigma$ is consumed entirely determines acceptance or rejection. It would be equivalent to require $(q_0, w, 0, \ldots, 0) \vdash_M^* (q, \lambda, c_1, \ldots, c_k)$, for some $q \in F$, but we continue to use \$ for compatibility with the end-marker definition. We use $\mathsf{DCM}_{\mathsf{NE}}(k, l)$ to denote the family of languages accepted by these machines when they have $k$ counters that are $l$-reversal-bounded. We define $\mathsf{DCM}_{\mathsf{NE}} = \bigcup_{k,l \geq 0} \mathsf{DCM}_{\mathsf{NE}}(k, l)$.

We denote by $\mathsf{2DCM}(1)$ to be the family of languages accepted by two-way deterministic finite automata (with both a left and right input tape end-marker) augmented by one reversal-bounded counter, accepted by final state. A machine of this form is said to be *finite-crossing* if there is a bound on the number of changes of direction on the input tape, and $t$-crossing if it makes at most $t$ changes of direction on the input tape for every computation.

## 3 Closure for Insertion and Concatenation Operations

Closure under concatenation is difficult for DCM languages because of determinism. However, we show special cases where closure results can be obtained. Additionally, we study the necessity of an end-of-tape marker, showing that it makes DCM languages strictly more powerful, but adding no power to $\mathsf{DCM}(1, l)$ languages. To our knowledge, the necessity of the right end-marker for one-way deterministic reversal-bounded multicounter machines has not been documented.

To show that the end-marker is not necessary for $\mathsf{DCM}(1, l)$, the proof of the lemma below takes an arbitrary $\mathsf{DCM}(1, l)$ machine $M$ and builds another $M'$ that accepts by final state without end-marker and accepts the same language. Before building $M'$, the construction builds an NCM machine for every state $q$ of $M$. This machine accepts all words of the form $a^i$ where there exists some word $x$ (this word is guessed using nondeterminism) such that $M$ can read $x$ from state $q$ and $i$ on the counter and reach a final state. Although these languages use nondeterminism, they are unary, and all NCM languages are semilinear [8], and all unary semilinear languages are regular [6]. Therefore, a DFA can be build for each such language (for each state of $M$). Because these languages are unary, the structure of the DFAs are well-known [12]. Every unary DFA is isomorphic to one with states $\{0, \dots, m-1\}$ where there exists some state $k$, and a transition from $i$ to $i+1$, for all $0 \leq i < k$ (the "tail"), and a transition from $j$ to $j+1$ for all $k \leq j < m-1$, plus a transition from $m$ to $k$ (the "loop"), and no other transitions. Let $t$ be the maximum tail size, over all DFAs constructed, plus one.

Then, intuitively, the construction of $M'$ involves $M'$ simulating $M$, and after reading input $w$, if $M$ has counter value $c$, $M'$ has counter value $c - t$ if $c > t$, with $t$ stored in the finite control. If $c \leq t$, then $M'$ stores $c$ in the finite control with zero on the counter. This allows $M'$ to know what counter value $M$ would have after reading a given word, but also to know when the counter value is less than $t$ (and the specific value less than $t$). In the finite control, $M'$ simulates each DFA in parallel. To do this, each time $M$ increases the counter, from $i$ to $i+1$, the state of each DFA switches forward by one letter. Each time $M$ decreases the counter from $i$ to $i-1$, the state of each DFA changes deterministically "going backwards in the loop" if $i > t$, and if $i \leq t$, then the counter of $M$ is stored in the finite control, and thus each DFA can tell when to switch deterministically from loop to tail. Then, when in state $q$ of $M$, $M'$ can tell if the current counter value would lead to acceptance from $q$ using the appropriate DFA.

The proof is omitted due to space constraints, and can be found online in [3].

**Lemma 3.** *For any $l$,* $\mathsf{DCM}(1, l) = \mathsf{DCM}_{\mathsf{NE}}(1, l)$.

We will extend these closure results with a lemma about prefix-free $\mathsf{DCM}_{\mathsf{NE}}$ languages. It was shown in [5] that a regular language is prefix-free if and only if there is a non-exiting DFA accepting the language. While we omit the proof (see [3]), the same logic gives this result for $\mathsf{DCM}_{\mathsf{NE}}$ languages.

**Lemma 4.** *Let $L \in \mathsf{DCM}_{\mathsf{NE}}$. Then $L$ is prefix-free if and only if there exists a* DCM*-machine $M$ accepting $L$ by final state without end-marker which is non-exiting.*

From this, we obtain a special case where DCM is closed under concatenation, if the first language can be both accepted by final state without end-marker, and is prefix-free. The construction considers a non-exiting machine accepting $L_1$ by final state without end-marker, where transitions into its final state are replaced by transitions into the initial state of the machine accepting $L_2$. The proof is omitted due to space constraints, and can be found online in [3].

**Proposition 5.** *Let* $L_1 \in \mathsf{DCM}_{\mathsf{NE}}(k,l), L_2 \in \mathsf{DCM}(k',l')$*, with* $L_1$ *prefix-free. Then* $L_1L_2 \in \mathsf{DCM}(k+k', \max(l,l'))$*.*

If we remove the condition that $L_1$ is prefix-free however, the proposition is no longer true, as we will see in the next section that even the regular language $\Sigma^*$ (which is in $\mathsf{DCM}_{\mathsf{NE}}(0,0)$) concatenated with a DCM language produces a language outside DCM.

**Corollary 6.** *Let* $L \in \mathsf{DCM}(k,l), R \in \mathsf{REG}$*, where* $R$ *is prefix-free. Then* $RL \in \mathsf{DCM}(k,l)$*.*

In contrast to left concatenation of a regular language with a DCM language (Corollary 6), where it is required that $R$ be prefix-free (the regular language is always in $\mathsf{DCM}_{\mathsf{NE}}$), for right concatenation, it is only required that it be a $\mathsf{DCM}_{\mathsf{NE}}$ language. We will see in the next section that this is not true if the restriction that $L$ accepts by final state without end-marker is removed.

The following proof takes a DCM machine $M_1$ accepting by final state without end-marker, and $M_2$ a DFA accepting $R$, and builds a DCM machine $M'$ accepting $LR$ by final state without end-marker. Intuitively, $M'$ simulates $M_1$ while also storing a subset of $M_2$'s states in a second component of the states. Every time it reaches a final state of $M_1$, it places the initial state of $M_2$ in the second component. Then, it continues to simulate $M_1$ while in parallel simulating the DFA $M_2$ separately on every state in the second component.

**Proposition 7.** *Let* $L \in \mathsf{DCM}_{\mathsf{NE}}(k,l)$*,* $R \in \mathsf{REG}$*. Then* $LR \in \mathsf{DCM}_{\mathsf{NE}}(k,l)$*. Also,* $\mathrm{pref}^{-1}(L) \in \mathsf{DCM}_{\mathsf{NE}}(k,l)$*.*

As a corollary, we get that $\mathsf{DCM}(1,l)$ is closed under right concatenation with regular languages. This corollary could also be inferred from the proof in [6] that deterministic context-free languages are closed under concatenation with regular languages.

**Corollary 8.** *Let* $L \in \mathsf{DCM}(1,l)$ *and* $R \in \mathsf{REG}$*. Then* $LR \in \mathsf{DCM}(1,l)$*.*

**Corollary 9.** *If* $L \in \mathsf{DCM}(1,l)$*, then* $\mathrm{pref}^{-1}(L) \in \mathsf{DCM}(1,l)$*.*

## 4 Relating (Un)Decidable Properties to Non-closure Properties

In this section, we use a technique that proves non-closure properties using (un)decidable properties. A similar technique was used in [2] for showing that there is a language accepted by a 1-reversal DPDA that cannot be accepted by any NCM. In particular, we use this technique to prove that some languages are not accepted by 2DCM(1)s (i.e., two-way DFAs with one reversal-bounded counter). Since 2DCM(1)s have two-way input and a reversal-bounded counter, it does not seem easy to derive "pumping" lemmas for these machines. 2DCM(1)s are quite powerful, e.g., although the Parikh map of the language accepted by any finite-crossing 2NCM (hence by any NCM) is semilinear [8], 2DCM(1)s can accept non-semilinear languages. For example, $L_1 = \{a^i b^k \mid i, k \geq 2, i \text{ divides } k\}$ can be accepted by a 2DCM(1) whose counter makes only one reversal. However, it is known that $L_2 = \{a^i b^j c^k \mid i, j, k \geq 2, k = ij\}$ cannot be accepted by a 2DCM(1) [9].

We will need the following result (the proof for DCMs is in [8]; the proof for 2DCM(1)s is in [9]):

**Theorem 10**

1. *The class of languages accepted by* DCM*s is closed under Boolean operations. Moreover, the emptiness problem is decidable.*
2. *The class of languages accepted by* 2DCM(1)*s is closed under Boolean operations. Moreover, the emptiness problem is decidable.*

We note that the emptiness problem for 2DCM(2)s, even when restricted to machines accepting only letter-bounded languages (i.e., subsets of $a_1^* \cdots a_k^*$ for some $k \geq 1$ and distinct symbols $a_1, \ldots, a_k$) is undecidable [8].

We will show that there is a language $L \in \mathsf{DCM}(1,1)$ such that $\inf^{-1}(L)$ is not in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$.

The proof uses the fact that that there is a recursively enumerable language $L_{\text{re}} \subseteq \mathbb{N}_0$ that is not recursive (i.e., not decidable) which is accepted by a deterministic 2-counter machine [11]. Thus, the machine when started with $n \in \mathbb{N}_0$ in the first counter and zero in the second counter, eventually halts (i.e., accepts $n \in L_{\text{re}}$).

A close look at the constructions in [11] of the 2-counter machine, where initially one counter has some value $d_1$ and the other counter is zero, reveals that the counters behave in a regular pattern. The 2-counter machine operates in phases in the following way. The machine's operation can be divided into phases, where each phase starts with one of the counters equal to some positive integer $d_i$ and the other counter equal to 0. During the phase, the positive counter decreases, while the other counter increases. The phase ends with the first counter having value 0 and the other counter having value $d_{i+1}$. Then in the next phase the modes of the counters are interchanged. Thus, a sequence of configurations corresponding to the phases will be of the form:

$$(q_1, d_1, 0), (q_2, 0, d_2), (q_3, d_3, 0), (q_4, 0, d_4), (q_5, d_5, 0), (q_6, 0, d_6), \ldots$$

where the $q_i$'s are states, with $q_1 = q_s$ (the initial state), and $d_1, d_2, d_3, \ldots$ are positive integers. Note that in going from state $q_i$ in phase $i$ to state $q_{i+1}$ in phase $i+1$, the 2-counter machine goes through intermediate states. Note that the second component of the configuration refers to the value of $c_1$ (first counter), while the third component refers to the value of $c_2$ (second counter).

For each $i$, there are 5 cases for the value of $d_{i+1}$ in terms of $d_i$: $d_{i+1} = d_i, 2d_i, 3d_i, d_i/2, d_i/3$. (The division operation is done only if the number is divisible by 2 or 3, respectively.) The case is determined by $q_i$. Thus, we can define a mapping $h$ such if $q_i$ is the state at the start of phase $i$, $d_{i+1} = h(q_i)d_i$ (where $h(q_i)$ is 1, 2, 3, 1/2, 1/3).

Let $T$ be a 2-counter machine accepting a recursively enumerable set $L_{\text{re}}$ that is not recursive. We assume that $q_1 = q_s$ is the initial state, which is never re-entered, and if $T$ halts, it does so in a unique state $q_h$. Let $T$'s state set be $Q$, and 1 be a new symbol.

In what follows, $\alpha$ is any sequence of the form $\#I_1\#I_2\#\cdots\#I_{2m}\#$ (thus we assume that the length is even), where $I_i = q1^k$ for some $q \in Q$ and $k \geq 1$, represents a possible configuration of $T$ at the beginning of phase $i$, where $q$ is the state and $k$ is the value of counter $c_1$ (resp., $c_2$) if $i$ is odd (resp., even).

Define $L_0$ to be the set of all strings $\alpha$ such that

1. $\alpha = \#I_1\#I_2\#\cdots\#I_{2m}\#$;
2. $m \geq 1$;
3. for $1 \leq j \leq 2m-1$, $I_j \Rightarrow I_{j+1}$, i.e., if $T$ begins in configuration $I_j$, then after one phase, $T$ is in configuration $I_{j+1}$ (i.e., $I_{j+1}$ is a valid successor of $I_j$);

**Lemma 11.** *$L_0$ is not in* $\mathsf{DCM} \cup \mathsf{2DCM}(1)$.

*Proof.* Suppose $L_0$ is accepted by a $\mathsf{DCM}$ (resp., $\mathsf{2DCM}(1)$). The following is an algorithm to decide, given any $n$, whether $n$ is in $L_{\text{re}}$.

1. Let $R = \#q_s1^n((\#Q1^+\#Q1^+))^*\#q_h1^+\#$. Clearly $R$ is regular.
2. Then $L' = L_0 \cap R$ is also in $\mathsf{DCM}$ (resp., $\mathsf{2DCM}(1)$) by Theorem 10.
3. Check if $L'$ is empty. This is possible, since emptiness of $\mathsf{DCM}$ (respectively, $\mathsf{2DCM}(1)$) is decidable by Theorem 10.

The claim follows, since $L'$ is empty if and only if $n$ is not in $L_{\text{re}}$. □

### 4.1 Non-closure Under Inverse Infix

**Theorem 12.** *There is a language $L \in \mathsf{DCM}(1,1)$ such that $\inf^{-1}(L)$ is not in* $\mathsf{DCM} \cup \mathsf{2DCM}(1)$.

*Proof.* Let $T$ be a 2-counter machine. Let $L = \{\#q1^m\#p1^n\# \mid T$ when started in state $q$ when one counter has value $m$ and the other counter has value 0, does not reach the configuration in the next phase where the first counter becomes zero, the other counter has value $n$, and the state is $p\}$. Thus, $L = \{\#I\#I'\# \mid I$ and $I'$ are configurations of $T$, and $I'$ is not a valid successor of $I\}$. Clearly, $L$ can be accepted by a $\mathsf{DCM}(1,1)$.

We claim that $L_1 = \inf^{-1}(L)$ is not in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. Otherwise, by Theorem 10, $\overline{L_1}$ (the complement of $L_1$) is also in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$, and $\overline{L_1} \cap (\#Q1^+\#Q1^+)^+\# = L_0$ would be in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. This contradicts Lemma 11. □

### 4.2 Non-closure Under Inverse Prefix

**Theorem 13.** *There exists a language $L$ such that $L \in \mathsf{DCM}(2,1)$ and $L \in \mathsf{2DCM}(1)$ (which makes only 1 turn on the input and 1 reversal on the counter) such that* $\text{pref}^{-1}(L) = L\Sigma^* \notin \mathsf{DCM} \cup \mathsf{2DCM}(1)$.

*Proof.* Consider $L = \{\#w\# \mid w \in \{a,b,\#\}^*, |w|_a \neq |w|_b\}$. Clearly, $L$ can be accepted by a DCM(2,1) and by a 2DCM(1) which makes only 1 turn on the input and 1 reversal on the counter.

Suppose to the contrary that $\text{pref}^{-1}(L) \in \mathsf{DCM} \cup \mathsf{2DCM}(1)$. Then, $L' \in \mathsf{DCM} \cup \mathsf{2DCM}(1)$, where $L' = \text{pref}^{-1}(L) \cap (\# \{a,b,\#\}^* \#) = \{\#w_1 \cdots \#w_n\# \mid \exists i.\, |w_1 \cdots w_i|_a \neq |w_1 \cdots w_i|_b\}$.

We know that $\mathsf{DCM}$ and $\mathsf{2DCM}(1)$ are closed under complement. So we can see that $L'' \in \mathsf{DCM} \cup \mathsf{2DCM}(1)$, where we define $L'' = \overline{L'} \cap (\#a^*b^*)^+\# = \{\#a^{k_1}b^{k_1}\# \cdots \#a^{k_m}b^{k_m}\# \mid m > 0\}$.

We will show that $L''$ is not in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. Suppose $L''$ is in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. Define two languages:

- $L_1 = \{\#1^{k_1}\#1^{k_1}\# \cdots \#1^{k_m}\#1^{k_m}\# \mid m \geq 1, k_i \geq 1\}$,
- $L_2 = \{\#1^{k_0}\#1^{k_1}\#1^{k_1}\# \cdots \#1^{k_{m-1}}\#1^{k_{m-1}}\#1^{k_m}\# \mid m \geq 1, k_i \geq 1\}$.

Note that $L_1$ and $L_2$ are similar. In $L_1$, the odd-even pairs of 1's are the same, but in $L_2$, the even-odd pairs of 1's are the same. Clearly, if $M''$ in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$ accepts $L''$, then we can construct (from $M''$) $M_1$ and $M_2$ in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$ to accept $L_1$ and $L_2$, respectively.

We now refer to the language $L_0$ that was shown not to be in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$ in Lemma 11. We will construct a DCM (resp., 2DCM(1)) to accept $L_0$, which would be a contradiction. Define the languages:

- $L_{odd} = \{\#I_1\#I_2\# \cdots \#I_{2m} \mid m \geq 1, I_1, \cdots, I_{2m}$ are configurations of the 2-counter machine $T$, for odd $i$, $I_{i+1}$ is a valid successor of $I_i\}$.
- $L_{even} = \{\#I_1\#I_2\# \cdots \#I_{2m} \mid m \geq 1, I_1, \cdots, I_{2m}$ are configurations of the 2-counter machine $T$, for even $i$, $I_{i+1}$ is a valid successor of $I_i\}$.

Clearly, $L_0 = L_{odd} \cap L_{even}$. Since DCM (resp., 2DCM(1)) is closed under intersection, we need only to construct two DCMs (resp., 2DCM(1)s) $M_{odd}$ and $M_{even}$ accepting $L_{odd}$ and $L_{even}$, respectively. We will only describe the construction of $M_{odd}$, the construction of $M_{even}$ being similar.

**Case:** Suppose $L'' \in \mathsf{DCM}$:
First consider the case of DCM. We will construct two machines: a DCM $A$ and a DFA $B$ such that $L(M_{odd}) = L(A) \cap L(B)$.

Let $L_A = \{\#I_1\#I_2\# \cdots \#I_{2m} \mid m \geq 1, I_1, \cdots, I_{2m}$ are configurations of the 2-counter machine $T$, for odd $i$, if $I_i = q_i 1^{d_i}$, then $d_{i+1} = h(q_i)d_i\}$. We can construct a DCM $A$ to accept $L_A$ by simulating the DCM $M_1$. For example, suppose $h(q_i) = 3$. Then $A$ simulates $M_1$ but whenever $M_1$ moves its input head one cell, $A$ moves its input head 3 cells. If $h(q_i) = 1/2$, then when $M_1$ moves its head 2 cells, $A$ moves its input head 1 cell. (Note that $A$ does not use the 2-counter machine $T$.)

Now Let $L_B = \{\#I_1\#I_2\#\cdots\#I_{2m} \mid m \geq 1, I_1, \cdots, I_{2m}$ are configurations of the 2-counter machine, for odd $i$, if $I_i = q_i 1^{d_i}$, then $T$ in configuration $I_i$ ends phase $i$ in state $q_{i+1}\}$. Clearly, a DFA $B$ can accept $L_B$ by simulating $T$ for each odd $i$ starting in state $q_i$ on $1^{d_i}$ *without* using a counter, and checking that the phase ends in state $q_{i+1}$. (Note that the DCM $A$ already checks the "correctness" of $d_{i+1}$.)

We can then construct from $A$ and $B$ a DCM $M_{odd}$ such that $L(M_{odd}) = L(A) \cap L(B)$. In a similar way, we can construct $M_{even}$.

**Case:** Suppose $L'' \in 2DCM(1)$:
The case 2DCM(1) can be shown similarly. For this case, the machines $M_{odd}$ and $M_{even}$ are 2DCM(1)s, and machine $A$ is a 2DCM(1), but machine $B$ is still a DFA. □

From this, we can immediately get the result that the right end-marker is necessary for deterministic counter machines when there are at least two 1-reversal-bounded counters. In fact, without it, no amount of reversal-bounded counters with a deterministic machine could accept even some languages that can be accepted with two 1-reversal-bounded counters could with the end-marker.

**Corollary 14.** *There are languages in* $\mathsf{DCM}(2,1)$ *that are not in* $\mathsf{DCM}_{\mathsf{NE}}$.

*Proof.* Since $\mathsf{DCM}_{\mathsf{NE}}$ is closed under concatenation with $\Sigma^*$, it follows that $\text{pref}^{-1}(L)$ from Theorem 13 is not in $\mathsf{DCM}_{\mathsf{NE}}$. □

### 4.3 Non-closure for Inverse Suffix, Outfix and Embedding

**Proposition 15.** *There exists a language* $L \in \mathsf{DCM}(1,1)$ *such that* $\text{suff}^{-1}(L) \notin \mathsf{DCM}$ *and* $\text{suff}^{-1}(L) \notin \mathsf{2DCM}(1)$.

*Proof.* Let $L$ be as in Theorem 12. We know $\mathsf{DCM}(1,1)$ is closed under $\text{pref}^{-1}$ by Corollary 9, so $\text{pref}^{-1}(L) \in \mathsf{DCM}(1,1)$. Suppose $\text{suff}^{-1}(\text{pref}^{-1}(L)) \in \mathsf{DCM}$. This implies that $\text{inf}^{-1}(L) \in \mathsf{DCM}$, but we showed this language was not in $\mathsf{DCM}$. Thus we have a contradiction. A similar contradiction can be reached when we assume $\text{suff}^{-1}(\text{pref}^{-1}(L)) \in \mathsf{2DCM}(1)$.

□

**Corollary 16.** *There exists* $L \in \mathsf{DCM}(1,1)$ *and regular languages* $R$ *such that* $RL \notin \mathsf{DCM}$ *and* $RL \notin \mathsf{2DCM}(1)$.

This implies that without the prefix-free condition on $L_1$ in Proposition 5, concatenation closure does not follow.

**Corollary 17.** *There exists* $L_1 \in \mathsf{DCM}_{\mathsf{NE}}(0,0)$ *(regular), and* $L_2 \in \mathsf{DCM}(1,1)$, *where* $L_1L_2 \notin \mathsf{DCM}$ *and* $L_1L_2 \notin \mathsf{2DCM}(1)$.

The result also holds for inverse outfix.

**Proposition 18.** *There exists a language* $L \in \mathsf{DCM}(1,1)$ *such that* $\text{outf}^{-1}(L) \notin \mathsf{DCM}$ *and* $\text{outf}^{-1}(L) \notin \mathsf{2DCM}(1)$.

*Proof.* Consider $L \subseteq \Sigma^*$ where $L \in \mathsf{DCM}(1,1)$, and $\text{suff}^{-1}(L) \notin \mathsf{DCM}$ and $\text{suff}^{-1}(L) \notin \mathsf{2DCM}(1)$. The existence of such a language is guaranteed by Proposition 15. Let $\Gamma = \Sigma \cup \{\%\}$.

Suppose $\text{outf}^{-1}(L) \in \mathsf{DCM}$. Then $L' \in \mathsf{DCM}$, where $L' = \text{outf}^{-1}(L) \cap \%\Sigma^*$. We can see $L' = \{\%yx \mid x \in L, y \in \Sigma^*\}$, since the language we intersected with ensures that the section is always added to the beginning of a word in $L$.

However, we also have $\%^{-1}L' \in \mathsf{DCM}$ because $\mathsf{DCM}$ is clearly closed under left quotient with a fixed word. We can see $\%^{-1}L' = \{yx \mid x \in L, y \in \Sigma^*\}$. This is just $\text{suff}^{-1}(L)$, so $\text{suff}^{-1}(L) \in \mathsf{DCM}$, a contradiction.

The result is the same for $\mathsf{2DCM}(1)$, relying on the closure of the family under left quotient with a fixed word, which is clear. ☐

**Corollary 19.** *Let* $m \in \mathbb{N}$. *There exists a language* $L \in \mathsf{DCM}(1,1)$ *such that* $\text{emb}^{-1}(m,L) \notin \mathsf{DCM}$ *and* $\text{emb}^{-1}(m,L) \notin \mathsf{2DCM}(1)$.

This is similar to Proposition 18 except starting with $\#^{m-1}$, then

$$\text{emb}^{-1}(\#^{m-1}L) \cap (\#\%)^{m-1}L = \{(\#\%)^{m-1}yx \mid x \in L, y \in \Sigma^*\},$$

and so $L' \in \mathsf{DCM}$.

## 5 Summary of Results

Assume $R \in \mathsf{REG}$, $L_{\mathsf{DCM}} \in \mathsf{DCM}$, and $L_{\mathsf{DCM_{NE}}} \in \mathsf{DCM_{NE}}$.

The question: For all $L \in \mathsf{DCM}(k,l)$:

**Table 1.** Summary of results for DCM. When applying the operation in the first column to any $L \in \mathsf{DCM}(k,l)$, is the result necessarily in $\mathsf{DCM}(k,l)$ (column 2), and in DCM (column 3)? This is parameterized in terms of $k$ and $l$, and the theorems showing each result is provided.

| **Operation** | is $Op(L) \in \mathsf{DCM}(k,l)$? | | is $Op(L) \in \mathsf{DCM}$? | |
|---|---|---|---|---|
| $\text{pref}^{-1}(L)$ | Yes if $k=1, l \geq 1$ | Cor 9 | Yes if $k=1, l \geq 1$ | Cor 9 |
| | No if $k \geq 2, l \geq 1$ | Thm 13 | Yes if $L \in \mathsf{DCM_{NE}}$ | Prop 7 |
| | | | No otherwise if $k \geq 2, l \geq 1$ | Thm 13 |
| $\text{suff}^{-1}(L)$ | No if $k, l \geq 1$ | Prop 15 | No if $k, l \geq 1$ | Prop 15 |
| $\text{inf}^{-1}(L)$ | No if $k, l \geq 1$ | Thm 12 | No if $k, l \geq 1$ | Thm 12 |
| $\text{outf}^{-1}(L)$ | No if $k, l \geq 1$ | Prop 18 | No if $k, l \geq 1$ | Prop 18 |
| $LR$ | Yes if $k=1, l \geq 1$ | Cor 8 | Yes if $k=1, l \geq 1$ | Cor 8 |
| | Yes if $L \in \mathsf{DCM_{NE}}$ | Prop 7 | Yes if $L \in \mathsf{DCM_{NE}}$ | Prop 7 |
| | No otherwise if $k \geq 2, l \geq 1$ | Thm 13 | No otherwise if $k \geq 2, l \geq 1$ | Thm 13 |
| $RL$ | Yes if $R$ prefix-free | Cor 6 | Yes if $R$ prefix-free | Cor 6 |
| | No otherwise if $k, l \geq 1$ | Cor 16 | No otherwise if $k, l \geq 1$ | Cor 16 |
| $L_{\mathsf{DCM}}L$ | No if $k, l \geq 1$ | Cor 17 | No if $k, l \geq 1$ | Cor 17 |
| $L_{\mathsf{DCM_{NE}}}L$ | No if $k, l \geq 1$ | Cor 17 | Yes if $L_{\mathsf{DCM_{NE}}}$ prefix-free | Prop 5 |
| | | | No otherwise if $k, l \geq 1$ | Cor 17 |

Also, for $\mathsf{2DCM}(1)$, the results are summarized as follows:

- There exists $L \in \mathsf{DCM}(1,1)$ (one-way), s.t. $\text{suff}^{-1}(L) \notin \mathsf{2DCM}(1)$ (Prop 15).
- There exists $L \in \mathsf{DCM}(1,1)$ (one-way) , $R$ regular, s.t. $RL \notin \mathsf{2DCM}(1)$ (Cor 16).
- There exists $L \in \mathsf{DCM}(1,1)$ (one-way), s.t. $\text{outf}^{-1}(L) \notin \mathsf{2DCM}(1)$ (Prop 18).
- There exists $L \in \mathsf{DCM}(1,1)$ (one-way), s.t. $\text{inf}^{-1}(L) \notin \mathsf{2DCM}(1)$ (Thm 12).
- There exists $L \in \mathsf{2DCM}(1)$, 1 input turn, 1 counter reversal, s.t. $\text{pref}^{-1}(L) \notin \mathsf{2DCM}(1)$ (Thm 13).
- There exists $L \in \mathsf{2DCM}(1)$, 1 input turn, 1 counter reversal, $R$ regular, s.t. $LR \notin \mathsf{2DCM}(1)$ (Thm 13).

This resolves every open question summarized above, optimally, in terms of the number of counters, reversals on counters, and reversals on the input tape.

## References

1. Baker, B.S., Book, R.V.: Reversal-bounded multipushdown machines. Journal of Computer and System Sciences **8**(3), 315–332 (1974)
2. Chiniforooshan, E., Daley, M., Ibarra, O.H., Kari, L., Seki, S.: One-reversal counter machines and multihead automata: Revisited. Theoretical Computer Science **454**, 81–87 (2012)
3. Eremondi, J., Ibarra, O., McQuillan, I.: Insertion operations on deterministic reversal-bounded counter machines. Tech. Rep. 2014–01, University of Saskatchewan (2014). http://www.cs.usask.ca/documents/techreports/2014/TR-2014-01.pdf
4. Gurari, E.M., Ibarra, O.H.: The complexity of decision problems for finite-turn multicounter machines. Journal of Computer and System Sciences **22**(2), 220–229 (1981)
5. Han, Y., Wood, D.: The generalization of generalized automata: Expression automata. International Journal of Foundations of Computer Science **16**(03), 499–510 (2005)
6. Harrison, M.: Introduction to Formal Language Theory. Addison-Wesley Pub. Co., Addison-Wesley series in computer science (1978)
7. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)
8. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. Journal of the ACM **25**(1), 116–133 (1978)
9. Ibarra, O.H., Jiang, T., Tran, N., Wang, H.: New decidability results concerning two-way counter machines. SIAM J. Comput. **23**(1), 123–137 (1995)
10. Jürgensen, H., Kari, L., Thierrin, G.: Morphisms preserving densities. International Journal of Computer Mathematics **78**, 165–189 (2001)
11. Minsky, M.L.: Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing Machines. Annals of Mathematics **74**(3), 437–455 (1961)
12. Nicaud, C.: Average state complexity of operations on unary automata. In: Kutyłowski, M., Pacholski, L., Wierzbicki, T. (eds.) Mathematical Foundations of Computer Science 1999. Lecture Notes in Computer Science, vol. 1672, pp. 231–240. Springer, Berlin Heidelberg (1999)