Insertion Operations on Deterministic Reversal-Bounded Counter Machines [*]

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

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

Ian McQuillan
Department of Computer Science, University of Saskatchewan
Saskatoon, SK S7N 5A9, Canada
`mcquillan@cs.usask.ca`

October 14, 2014

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, a new technique is created 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.

[*]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.

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 machine is reversal-bounded [1, 7]. 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 [7]. 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 [3].

Reversal-bounded counter machines (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 $w \leq_{\mathrm{i}} z$, if $z = xwy$, for some $x, y \in \Sigma^*$. Viewed as an operation on the first component of the relation, $\leq_{\mathrm{i}} (w) = \{z \mid w \leq_{\mathrm{i}} z, z \in \Sigma^*\}$, 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, $z \leq_{\mathrm{i}}^{-1} w$, if $z = xwy$, then viewing this as an operation, $\leq_{\mathrm{i}}^{-1} (z) = \{w \mid z \leq_{\mathrm{i}}^{-1} w, w \in \Sigma^*\} = \{w \mid w \leq_{\mathrm{i}} z\}$, the set of all infixes of z. These can be extended to operations on languages. Viewed in this way, we can define the prefix, suffix, infix and outfix operations, along with their inverses on languages. This is the approach taken in [9]. Using the more common notation of $\mathrm{inf}(L)$ for the set of infixes of L, then $\mathrm{inf}^{-1}(L) = \Sigma^* L \Sigma^*$, 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* as done in [9]. 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 still 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 Σ^* 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 [5]. In addition, concatenating Σ^* 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 depend on a new technique developed herein to simultaneously show languages are not in DCM and not in 2DCM(1). The technique does not rely on any pumping arguments (like the recent paper [2] that developed a technique to show languages are not in DCM) as the latter family even contains non-semilinear languages.

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 [5, 6]. The empty word is denoted by λ. We use Σ and Γ to represent finite alphabets, with Σ^* as the set of all words over Σ and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. For a word $w \in \Sigma^*$, if $w = a_1 \cdots a_n$ where $a_i \in \Sigma$, $1 \leq i \leq n$, 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:*

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

We generalize the outfix relation to the notion of embedding [9]:

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

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

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, respectively, $Q, \Sigma, \$, q_0, F$ are 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 δ (defined as in [7] 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 $d_i \ge 0$ 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 δ 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 language families are denoted by $\mathsf{NCM} = \bigcup_{k,l \ge 0} \mathsf{DCM}(k,l)$ and $\mathsf{DCM} = \bigcup_{k,l \ge 0} \mathsf{DCM}(k,l)$.

Given a DCM machine $M = (k, Q, \Sigma, \$, \delta, q_0, F)$, the language accepted by *final state without end-marker* is the set of all 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 $q \in F$, $q' \in Q$, $a \in \Sigma$, $c_i, c_i' \in \mathbb{N}_0, 1 \le i \le k$. 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 Σ 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 the restriction of being deterministic. 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 in the literature.

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

Proof. By definition, $\mathsf{DCM}_{\mathsf{NE}}(1, l) \subseteq \mathsf{DCM}(1, l)$.

Consider $M = (1, Q, \Sigma, \$, \delta, q_0, F)$ accepting L by final state. A machine M' will be built such that the language accepted by M' by final state without end-marker is equal to $L(M)$.

We assume without loss of generality that δ is defined for all inputs. Let $|Q| = n$. For each state $q \in Q$, we can define the language

$$L(q) = \{a^i \mid \exists w \text{ such that } (q, w\$, i) \vdash_M^* (q_f, \$, c), q_f \in F, c \in \mathbb{N}_0\},$$

the set of counter values which leads to acceptance on end-of-tape marker \$ from state q. Since this language is in NCM, NCM languages are semilinear, and $L(q)$ is unary, we know $L(q)$ is regular. Thus we can accept $L(q)$ with a DFA, say $D(q) = (Q_{D(q)}, \{a\}, \delta_{D(q)}, s_{D(q)}, F_{D(q)})$. We denote by $D(q, i)$ the state of $D(q)$ reached after reading i letters of input.

Because these languages are unary, the structure of the DFAs are relatively simple, and well-known [11]. Every unary DFA with m states is isomorphic to one with states $\{0, \ldots, m-1\}$ where there exists some state k, and there is a transition from from i to $i+1$, for all $0 \leq i < k$ (the "tail"), and there is 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. Also, assume without loss of generality that there is always a non-empty loop, which can be assumed by adding a loop without any final states. Thus, $D(q,i)$ is always defined for every $i \geq 0$.

Let t be one more than the maximum tail size of any $D(q)$. Then $t > 0$. The intuition for the construction of M' is as follows. The machine M' simulates M, and after reading 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 $D(q)$ in parallel in such a way that the state $D(q,i)$ is stored, for each $q \in Q$, when the counter of M is i. To do this, each time M increases the counter, from i to $i+1$, the state of each $D(q)$ switches from $D(q,i)$ to $D(q,i+1)$. Each time M decreases the counter from i to $i-1$, the state of each $D(q)$ 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 $M(q)$ 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 be accepted using the appropriate DFA $D(q)$.

We now provide the construction in detail:

The machine M' has state set $Q_{M'} = Q \times \{0, \ldots, t\} \times Q_{D(q_1)} \times \cdots \times Q_{D(q_n)}$, where $Q = \{q_1, \ldots, q_n\}$ (and so q_0 is also q_i, for some $i, 1 \leq i \leq n$).

A state of M' is final if it is of the form $(q_i, c, d_1, \ldots, d_i, \ldots, d_n)$ where d_i is final in $D(q_i)$, for any $1 \leq i \leq n$. The initial state is $(q_0, 0, D(q_1, 0), \ldots, D(q_n, 0))$.

For each state $d \in Q_{D(q_i)}$ for each q_i, we define $\mathit{next}(d) = \delta_{D(q_i)}(d, a)$. We define $\mathit{prev}(d)$ as the unique state d' where $\delta_{D(q_i)}(d', a) = d$ and d' is in the loop of $D(q_i)$, defined only if d is in the loop.

If $\delta_M(q, b, 0) = (p, T, i)$, for some $q, p \in Q, b \in \Sigma, T \in \{\mathrm{S}, \mathrm{R}\}, i \in \{0, 1\}$, we add the following transition to $\delta_{M'}$:

1. $\delta_{M'}((q, 0, D(q_1, 0), \ldots, D(q_n, 0)), b, 0) = (p, i, D(q_1, i), \ldots, D(q_n, i), T, 0)$.

Also, if $\delta_M(q, b, 1) = (p, T, i)$, for some $q, p \in Q, T \in \{\mathrm{S}, \mathrm{R}\}, i \in \{-1, 0, 1\}$ we add the following transition in $\delta_{M'}$ for every $s = (q, c, d_1, \ldots, d_n)$:

2. $\delta(s, b, 0) = ((p, c+i, D(q_1, c+i), \ldots, D(q_n, c+i), T, 0)$ if either $0 < c < t$ or $c = t$ and $i \in \{0, -1\}$,

3. $\delta(s, b, 1) = ((p, t, d_1, \ldots, d_n)), T, 0)$, if $c = t$ and $i = 0$,

4. $\delta(s, b, x) = ((p, t, \mathit{next}(d_1), \ldots, \mathit{next}(d_n)), T, i)$ for $x \in \{0, 1\}$, if $c = t$ and $i = 1$,

5. $\delta(s, b, 1) = ((p, t, \mathit{prev}(d_1), \ldots, \mathit{prev}(d_n)), T, i)$ if $c = t$ and $i = -1$.

We can see that for any counter value c and M-state q, $next(D(q,c)) = D(q,c+1)$. Additionally, if $c > t$, then $c-1 \geq t$, meaning that $D(q,c-1)$ must be in the loop of the unary DFA $D(q)$. Thus, $prev(D(q,c)) = D(q,c-1)$.

Claim 1. *Let $w \in \Sigma^*$. For all $m \in \mathbb{N}_0$, if $(q_0, w = uv, 0) \vdash_M^m (q, v, c)$ for some $u, v \in \Sigma^*$, then*

$$((q_0, 0, D(q_1,0), \ldots, D(q_n,0)), uv, 0) \vdash_{M'}^m ((q, t, D(q_1,c), \ldots, D(q_n,c)), v, c-t),$$

when $c > t$, and

$$((q_0, 0, D(q_1,0), \ldots, D(q_n,0)), uv, 0) \vdash_{M'}^m ((q, c, D(q_1,c), \ldots, D(q_n,c)), v, 0),$$

when $c \leq t$.

Proof. We perform induction on m.

If $m = 0$ then $q = q_0, u = \lambda, c = 0, c \leq t$, thus the second condition is true.

Consider $m \geq 0$, and assume the implication holds for m. We will show it holds for $m+1$.

Suppose $(q_0, uv, 0) \vdash_M^{m+1} (q, v, c)$. Then for some state $p \in Q$, $a \in \Sigma \cup \{\lambda\}$, and $c' \in \mathbb{N}_0$, we have $(q_0, uv, 0) \vdash_M^m (p, av, c') \vdash_M^1 (q, v, c)$. We know that $c \in \{c'-1, c', c'+1\}$.

Case: $c > t, c' > t$. By our hypothesis, we have

$$((q_0, 0, D(q_1,0), \ldots, D(q_n)), uv, 0) \vdash_{M'}^m ((p, t, D(q_1,c'), \ldots, D(q_n,c')), av, c'-t).$$

If $c = c'-1$, then we know that $((p, t, D(q_1,c'), \ldots, D(q_n,c')), av, c'-t) \vdash_{M'} ((q, t, D(q_1,c), \ldots, D(q_n,c)), v, c-t)$ by transition rule (5).

If $c = c'$, then we know that

$$((p, t, D(q_1,c'), \ldots, D(q_n,c')), av, c'-t) \vdash_{M'} ((q, t, D(q_1,c), \ldots, D(q_n,c)), v, c-t)$$

by transition rule (3).

If $c = c'+1$, then we know

$$((p, t, D(q_1,c'), \ldots, D(q_n,c')), av, c'-t) \vdash_{M'} ((q, t, D(q_1,c), \ldots, D(q_n,c)), v, c-t)$$

by transition rule (4).

Case: $c > t, c' \leq t$. Then we know that $c' = t$ and $c = t+1$. By our hypothesis, we have:

$$((q_0, 0, D(q_1,0), \ldots, D(q_n,0)), uv, 0) \vdash_{M'}^m ((p, t, D(q_1,t), \ldots, D(q_n,t)), av, 0),$$

$a \in \Sigma \cup \{\lambda\}$. So by transition rule (4) we have

$$((p, t, D(q_1,t), \ldots, D(q_n,t)), av, 0) \vdash_{M'} ((q, t, D(q_1,t+1), \ldots, D(q_n,t+1)), v, 1),$$

which is valid since $1 = c - c' = c - t$.

Case: $c \leq t, c' \leq t$. By our hypothesis we have

$$((q_0, 0, D(q_1, 0), \ldots, D(q_n, 0)), uv, 0) \vdash_{M'}^{m} ((p, c', D(q_1, c'), \ldots, D(q_n, c')), av, 0),$$

$a \in \Sigma \cup \{\lambda\}$.

If $c = c' - 1$, $c = c'$ or $c = c' + 1$, then the implication holds by transition rule (2), unless $c' = 0$, in which case it holds by transition rule (1).

Case: $c \leq t, c' > t$. Then we know $c = t$ and $c' = t + 1$. By our hypothesis we have $((q_0, 0, D(q_1, 0), \ldots, D(q_n, 0)), uv, 0) \vdash_{M'}^{m} ((p, t, D(q_1, t+1), \ldots, D(q_n, t+1)), av, 1)$, for some $a \in \Sigma \cup \{\lambda\}$. The implication then holds from transition rule (5), since $c = t$ and t is one more than the largest tail, which means $D(q_j, t)$ is still in the loop, for every j, $1 \leq j \leq n$.

Thus we have shown that the implication true for M' in $m + 1$ steps, and is therefore true for all m. □

Claim 2. *Let $w \in \Sigma^*$. For all $m \in \mathbb{N}_0$, if*

$$((q_0, 0, D(q_1, 0), \ldots, D(q_n, 0)), w = uv, 0) \vdash_{M'}^{m} ((q, d, q'_1, \ldots, q'_n), v, e)$$

and c is the number of transitions used in the derivation where $i = 1$ in the construction, minus the number of transitions used where $i = -1$ in the construction, then the following are true:

- *$q'_j = D(q_j, c)$, for $1 \leq j \leq n$,*
- *$d = t, e = c - t$ when $c > t$,*
- *$d = c, e = 0$ when $c \leq t$,*
- *$(q_0, w = uv, 0) \vdash_{M}^{m} (q, v, c)$.*

Proof. We perform induction on m.

If $m = 0$ then $q = q_0, u = \lambda, c = 0 < t, d = e = 0$, and the conclusion is true.

Suppose $((q_0, 0, d(q_1, 0), \ldots, d(q_n)), uv, 0) \vdash_{M'}^{m+1} ((q, d, q'_1, \ldots, q'_n), v, e)$. Then

$$\begin{aligned} ((q_0, 0, d(q_1, 0), \ldots, d(q_n)), uv, 0) &\vdash_{M'}^{m} ((q', d', q''_1, \ldots, q''_n), av, e') \\ &\vdash_{M} ((q, d, q'_1, \ldots, q'_n), v, e), \end{aligned}$$

$a \in \Sigma \cup \{\lambda\}$ by some transition t. Let c' be the number of transitions used in the first m transitions of the derivation where $i = 1$, minus those where $i = -1$. Then, by the hypothesis,

- $((q_0, 0, d(q_1, 0), \ldots, d(q_n)), uv, 0) \vdash_{M'}^{m} ((q', d', D(q_1, c'), \ldots, D(q_n, c'), av, e')$,
- $d' = t, e' = c' - t$, when $c' > t$, and

- $d' = c', e' = 0$, when $c' \leq t$ and
- $(q_0, uv, 0) \vdash_M^m (q', av, c')$.

Suppose that $c' > t$. Then

$$\begin{aligned}((q_0, 0, d(q_1, 0), \ldots, d(q_n)), uv, 0) &\vdash_{M'}^m ((q', t, D(q_1, c'), \ldots, D(q_n, c')), av, c' - t) \\ &\vdash_M ((q, d, q_1', \ldots, q_n'), v, e),\end{aligned}$$

$a \in \Sigma \cup \{\lambda\}$ by some transition t. If t is created from a transition where $i = 1$, this must be of type (4), and let $c = c' + 1$ then $q_j' = D(q_j, c), 1 \leq j \leq n, d = t, e = c - t$, and thus $(q_0, uv, 0) \vdash_M^* (q, v, c)$. If t is created from a transition where $i = 0$, this must be of type (4), $c = c', q_j' = D(q_j, c), 1 \leq j \leq n, d = t, e = c - t$, and thus $(q_0, uv, 0) \vdash_M^* (q, v, c)$. If t is created from a transition where $i = -1$, then t must be of type (5), and let $c = c' - 1$ then $c \geq t, q_j' = D(q_j, c), 1 \leq j \leq n, d = t, e = c - t$ (and so $d = c$ and $e = 0$ if $c = t$), and thus $(q_0, uv, 0) \vdash_M^* (q, v, c)$.

If $c' \leq t$, the proof is also similar. □

As a result of this claim, we can see that, $(q_0, w\$, 0) \vdash_M^* (q, \$, c)$, where $(q, \$, c)$ is the first configuration reaching \$, if and only if

$$((q_0, 0, D(q_1, 0), \ldots, D(q_n, 0)), w, 0) \vdash_{M'}^* ((q, t, D(q_1, c), \ldots, D(q_n, c)), \lambda, c - t)$$

if $c > t$ and

$$((q_0, 0, D(q_1, 0), \ldots, D(q_n, 0)), w, 0) \vdash_{M'}^* ((q, c, D(q_1, c), \ldots, D(q_n, c)), \lambda, 0)$$

if $c \leq t$. Then, M accepts w if and only if $a^c \in D(q)$, by our definition of $D(q)$. And M' accepts w if and only if $D(q, c)$ is final in $D(q)$, which again happens if and only if $a^c \in \mathsf{D}(q)$. So M accepts w by final state if and only if M' accepts w by final state without end-marker. □

We will extend these closure results with a lemma about prefix-free $\mathsf{DCM_{NE}}$ languages. It was shown in [4] that a regular language is prefix-free if and only if there is a non-exiting DFA (defined just like with counter machines above, where there are no transitions defined on final states) accepting the language.

Lemma 2. *Let $L \in \mathsf{DCM_{NE}}$. 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.*

Proof. Let $L \in \mathsf{DCM_{NE}}$, with M a machine accepting L by final state without end-marker.

$(\implies)$ Suppose L is prefix-free. Construct M' from M such that all transitions defined on final states are removed, and so M' is not non-exiting. Then $L(M') \subseteq L(M)$ since all transitions of M' are in M. Let q_f be a final state with outgoing transitions in M. For every w leading to state q_f in M, wx is not accepted for any x.

Then, we can accept $w \in M'$ since M accepts L by final state without end-marker, thus removing transitions of q_f that will not disrupt the acceptance of w in M'. Thus $L(M) \subseteq L(M')$ as well.

($\Longleftarrow$) Suppose M is non-exiting accepting L by final state without end-marker. Consider $w \in L$. Then after reading w, there are no transitions to follow, so wx is not accepted for any x. Thus L is prefix-free.

$\square$

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.

Proposition 1. *Let* $L_1 \in \mathsf{DCM}_{\mathsf{NE}}(k,l), L_2 \in \mathsf{DCM}(k',l')$*, with* L_1 *prefix-free. Then* $L_1L_2 \in \mathsf{DCM}(k+k', \max(l,l'))$*.*

Proof. Our construction is simple. Consider non-exiting M_1 accepting L_1 by final state without end-marker, and M_2 accepting L_2. We form M' where $L(M') = L_1L_2$. Indeed, M' has the states and transitions from M_1, M_2 combined, with the start state of M_1 as its start state. Any transition into an accepting state of M_1 is replaced by an equivalent transition into the starting state of M_2. The accepting states are the accepting states of M_2. The machine has separate counters for the counters of M_1 and M_2, each of which performs the same reversals they would in their original machine.

If $w \in L_1$ and $x \in L_2$, then $wx \in L(M')$. Since M_1 accepts without end-marker, we know reading w in M_1 immediately leads to an accepting state in M_1, even without reading \$. So, in M', we know that reading w will lead to the start state of M_2. Reading x from the start of M_2 leads to an accepting state, since $x \in L_2$. Thus reading x from the start of M_2 in M' leads to acceptance.

If $y \in L(M')$, then reading y lead to some accepting state of M_2. M' starts in the start of M_1, so the only path to an accepting state is through the start of M_2. Thus there is some division of y into w, x where reading w in M_1 leads to acceptance (because it leads to the start state of M_2 in M'), and reading x in M_2 leads to acceptance, because we got to an accepting state in M'. Thus $y \in L_1L_2$. $\square$

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 Σ^* (which is in $\mathsf{DCM}_{\mathsf{NE}}(0,0)$) concatenated with a DCM language produces a language outside DCM.

Corollary 1. *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 1), 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.

Proposition 2. *Let $L \in \mathsf{DCM}_{\mathsf{NE}}(k,l)$, $R \in \mathsf{REG}$. Then $LR \in \mathsf{DCM}_{\mathsf{NE}}(k,l)$. Also,* $\text{pref}^{-1}(L) \in \mathsf{DCM}_{\mathsf{NE}}(k,l)$.

Proof. Let $M_1 = (k, Q_1, \Sigma, \$, \delta_1, q_1, F_1)$ be a DCM machine accepting L by final state without end-marker. Let $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be a DFA accepting R. A DCM machine M' will be built that will accept LR by final state without end-marker. Assume without loss of generality that M_1 reads all the way to the end of every input.

Intuitively, $M' = (k, Q', \Sigma, \$, \delta', q', F')$ will simulate M_1 while also storing a subset of Q_2 in the second component of the state. Every time it reaches a final state of M_1, it places the initial state of M_2 in the second component. And, then it continues to simulate M_1, while in parallel simulating the DFA M_2 on every state in the second component in parallel.

Formally, $Q' = Q_1 \times 2^{Q_2}$, $q' = (q_1, \emptyset)$ if $q_1 \notin F_1$ and $q' = (q_1, \{q_2\})$ otherwise, $F' = \{(q, X) \mid q \in Q, X \cap F_2 \neq \emptyset\}$ and δ' is defined as follows: For every transition, $\delta_1(q, a, x) = (p, T, i_1, \dots, i_k), p, q \in Q, a \in \Sigma, x \in \{0,1\}, T \in \{\text{S}, \text{R}\}, i_j \in \{-1, 0, 1\}, 1 \le j \le k$, introduce $\delta'((q, Y), a, x) = ((p, Z), T, i_1, \dots, i_k)$, for all $Y \in 2^{Q_2}$, where

1. $Z = Y$ if $T = \text{S}$ and $p \notin F_1$,
2. $Z = \delta_2(Y, a)$ if $T = \text{R}$ and $p \notin F_1$,
3. $Z = Y \cup \{q_2\}$ if $T = \text{S}$ and $p \in F_1$,
4. $Z = \delta_2(Y, a) \cup \{q_2\}$ if $T = \text{R}$ and $p \in F_1$.

Claim 3. *$L(M_1)L(M_2) \subseteq L(M')$.*

Proof. Let $uv \in \Sigma^*$, where $u \in L(M_1), v \in L(M_2)$. Then there is a derivation $(p_1, u_1, i_1(1), \dots, i_k(1)) \vdash_M \cdots \vdash_M (p_n, u_n, i_1(n), \dots, i_k(n))$ where $p_1 = q_1, u_1 = uv, i_1(1) = \cdots = i_k(1) = 0, p_n \in F_1, u_n = v$, and the last transition applied reads a letter (moves right) since M_1 accepts by final state without end-marker. Furthermore, since M_1 reads every input, $(p_n, u_n, i_1(n), \dots, i_k(n)) \vdash_{M_1}^* (p', \lambda, i_1, \dots, i_k)$ for some $p' \in Q_1, i_j \in \mathbb{N}_0, 1 \le j \le k$. So by the construction $((q_1, Y_1), uv, 0, \dots, 0) \vdash_{M'}^* ((p_n, Y_n), v, i_1(n), \dots, i_k(n))$, either this sequence is of length 0, or longer where the last transition applied reads a letter, and since $p_n \in F_1$, the last transition applied was of type 4 above. In both cases, $q_2 \in Y_n$. In addition, it must be the case that

$$((p_n, Y_n), v, i_1(n), \dots, i_k(n)) \vdash_{M'}^* ((p', Y'), \lambda, i_1, \dots, i_k),$$

and that $\hat{\delta}(q_2, v) \in Y'$ since every transition applied to M_1 while reading v that consumes an input letter, also changes state via that letter according to the DFA M_2. Thus, there is a final state from F_2 in Y' causing M' to also accept. $\square$

Claim 4. $L(M') \subseteq L(M_1)L(M_2)$

Proof. Let $w \in L(M')$. Then,

$$((q_1, Y_1), w_1, i_1(1), \ldots, i_k(1)) \vdash_{M'} \cdots \vdash_{M'} ((p_n, Y_n), w_n, i_1(n), \ldots, i_k(n)),$$

where $w_1 = w, q' = (q_1, Y_1), i_1(1) = \cdots = i_k(1) = 0, Y_n \cap F_2 \neq \emptyset, w_n = \lambda$. Let q_f be some state in $F_2 \cap Y_n$. Then, by the construction, there exists some j, $1 \le j \le n$ such that $p_j \in F_1, q_2 \in Y_j$, and for every transition from the jth configuration to the last one, while reading w_j, the sets $Y_j, \ldots, Y_n$ consecutively stay the same on a stay transition, and on a right transition that consumes the next input letter of w_j, puts the state $\hat{\delta_2}(q_2, w'_j)$, for each consecutive prefix of w'_j of w_j in the sets $Y_j, \ldots, Y_n$. Hence, $w_j \in R$, and since $p_j \in F_1$, it must be that $ww_j^{-1} \in L(M_1)$. $\square$

Hence, $LR \in \mathsf{DCM}_{\mathsf{NE}}(k, l)$. $\square$

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 [5] that deterministic context-free languages are closed under concatenation with regular languages.

Corollary 2. *Let* $L \in \mathsf{DCM}(1, l)$ *and* $R \in \mathsf{REG}$. *Then* $LR \in \mathsf{DCM}(1, l)$.

Corollary 3. *If* $L \in \mathsf{DCM}(1, l)$, *then* $\text{pref}^{-1}(L) \in \mathsf{DCM}(1, l)$.

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

In this section, we demonstrate a technique that proves non-closure properties using (un)decidable properties. 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 [7], 2DCM(1)s can accept non-semilinear languages. For example, $L_1 = \{a^i b^k \mid i, k \ge 2, i \text{ divides } k\}$ can be accepted by a 2DCM(1) whose counter makes only one reversal. However, it is known that $L_2 = \{a^i b^j c^k \mid i, j, k \ge 2, k = ij\}$ cannot be accepted by a 2DCM(1) [8].

We will need the following result (the proof for DCMs is in [7]; the proof for 2DCM(1)s is in [8]):

Theorem 1.

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* $2\mathsf{DCM}(1)$*s is closed under Boolean operations. Moreover, the emptiness problem is decidable.*

We note that the emptiness problem for $2\mathsf{DCM}(2)$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 [7].

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

The proof uses the fact that that there is a recursively enumerable language $L_{\rm re} \subseteq \mathbb{N}_0$ that is not recursive (i.e., not decidable) which is accepted by a detereminstic 2-counter machine [10]. 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 $n \in L_{\rm re}$).

A close look at the contructions in [10] 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.

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).

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).

Let T be a 2-counter machine accepting a recursively enumberable set $L_{\rm re}$ 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, α 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 α 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 3. *L_0 is not in* $\mathsf{DCM} \cup \mathsf{2DCM}(1)$.

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

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

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

4.1 Non-closure Under Inverse Infix

Theorem 2. *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 counter become 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 $\mathsf{DCM}(1,1)$.

Let Σ be the alphabet over which L is defined. We claim that $L_1 = \inf^{-1}(L)$ is not in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. Otherwise, by Theorem 1, $\overline{L_1}$ (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 3. $\square$

4.2 Non-closure Under Inverse Prefix

Theorem 3. *There exists a language L such that $L \in \mathsf{DCM}(2,1)$ and $L \in 2\mathsf{DCM}(1)$ (which makes only 1 turn on the input and 1 reversal on the counter) such that* $\mathrm{pref}^{-1}(L) = L\Sigma^* \notin \mathsf{DCM} \cup 2\mathsf{DCM}(1)$.

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 $\mathrm{pref}^{-1}(L) \in \mathsf{DCM} \cup 2\mathsf{DCM}(1)$. Then, $L' \in \mathsf{DCM} \cup 2\mathsf{DCM}(1)$, where

$$L' = \mathrm{pref}^{-1}(L) \cap (\#\{a,b,\#\}^*\#) = \{\#w_1 \cdots \#w_n\# \mid \exists i.\, |w_1 \cdots w_i|_a \neq |w_1 \cdots w_i|_b\}.$$

We know that DCM and $2\mathsf{DCM}(1)$ are closed under complement. So we can see that $L'' \in \mathsf{DCM} \cup 2\mathsf{DCM}(1)$, where we define

$$L'' = \overline{L'} \cap (\#a^*b^*)^+\# = \left\{\#a^{k_1}b^{k_1}\# \cdots \#a^{k_m}b^{k_m}\# \mid m > 0\right\}.$$

We will show that L'' is not in $\mathsf{DCM} \cup 2\mathsf{DCM}(1)$. Suppose L'' is in $\mathsf{DCM} \cup 2\mathsf{DCM}(1)$. 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 $\mathsf{DCM} \cup 2\mathsf{DCM}(1)$ accepts L'', then we can construct (from M'') M_1 and M_2 in $\mathsf{DCM} \cup 2\mathsf{DCM}(1)$ to accept L_1 and L_2, respectively.

We now refer to the language L_0 that was shown not to be in $\mathsf{DCM} \cup 2\mathsf{DCM}(1)$ in Lemma 3. 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 (rersp., 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: L'' not in 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: L'' not 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 4. *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 Σ^*, it follows that $\text{pref}^{-1}(L)$ from Theorem 3 is not in $\mathsf{DCM}_{\mathsf{NE}}$. □

4.3 Non-closure for Inverse Suffix, Outfix and Embedding

Proposition 3. *There exists a language in* $L \in \mathsf{DCM}(1,1)$ *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 2. We know $\mathsf{DCM}(1,1)$ is closed under pref^{-1} by Corollary 3, 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 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 5. *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 1, concatenation closure does not follow.

Corollary 6. *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 4. *There exists a language $L \in \mathsf{DCM}(1,1)$ such that $\mathrm{outf}^{-1}(L) \notin \mathsf{DCM}$ and $\mathrm{outf}^{-1}(L) \notin \mathsf{2DCM}(1)$.*

Proof. Consider $L \subseteq \Sigma^*$ where $L \in \mathsf{DCM}(1,1)$, $\mathrm{suff}^{-1}(L) \notin \mathsf{DCM}$ and $\mathrm{suff}^{-1}(L) \notin \mathsf{2DCM}(1)$. The existence of such a language is guaranteed by Proposition 3. Let $\Gamma = \Sigma \cup \{\%\}$.

Suppose $\mathrm{outf}^{-1}(L) \in \mathsf{DCM}$. Then $L' \in \mathsf{DCM}$, where $L' = \mathrm{outf}^{-1}(L) \cap \%\Sigma^*$. 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 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 $\mathrm{suff}^{-1}(L)$, so $\mathrm{suff}^{-1}(L) \in \mathsf{DCM}$, 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. $\square$

Corollary 7. *Let $m \in \mathbb{N}$. There exists a language $L \in \mathsf{DCM}(1,1)$ such that $\mathrm{emb}^{-1}(m, L) \notin \mathsf{DCM}$ and $\mathrm{emb}^{-1}(m, L) \notin \mathsf{2DCM}(1)$.*

This is similar to Proposition 4 except starting with $\#^{m-1}$, then

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

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

5 Summary of Results

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

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

Operation	is $Op(L) \in \mathsf{DCM}(k,l)$?	is $Op(L) \in \mathsf{DCM}$?
$\text{pref}^{-1}(L)$	Yes $\forall l, \mathsf{DCM}(1,l)$ No $\forall k \geq 2, l \geq 1$	Yes for $\mathsf{DCM}(1,l)$ Yes $L \in \mathsf{DCM}_{\mathsf{NE}}$ No otherwise, $\forall k \geq 2, l \geq 1$
$\text{suff}^{-1}(L)$	No $\forall k, l \geq 1$	No $\forall k, l \geq 1$
$\text{inf}^{-1}(L)$	No $\forall k, l \geq 1$	No $\forall k, l \geq 1$
$\text{outf}^{-1}(L)$	No $\forall k, l \geq 1$	No $\forall k, l \geq 1$
LR	Yes $\forall \mathsf{DCM}(1,l)$ Yes if $L \in \mathsf{DCM}_{\mathsf{NE}}$ No otherwise, $\forall k \geq 2, l \geq 1$	Yes $\forall l, \mathsf{DCM}(1,l)$ Yes if $L \in \mathsf{DCM}_{\mathsf{NE}}$ No otherwise, $\forall k \geq 2, l \geq 1$
RL	Yes if R prefix-free No otherwise, $\forall k, l \geq 1$	Yes if R prefix-free No otherwise, $\forall k, l \geq 1$
$L_{\mathsf{DCM}}L$	No $\forall k, l \geq 1$	No $\forall k, l \geq 1$
$L_{\mathsf{DCM}_{\mathsf{NE}}}L$	No $\forall k, l \geq 1$	Yes if $L_{\mathsf{DCM}_{\mathsf{NE}}}$ prefix-free No $\forall k, l \geq 1$ if not prefix-free

Table 1: Summary of results for DCM.

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

- There exists $L \in \mathsf{DCM}(1,1)$ (one-way), such that $\text{suff}^{-1}(L) \notin 2\mathsf{DCM}(1)$.
- There exists $L \in \mathsf{DCM}(1,1)$ (one-way) , R regular, such that $RL \notin 2\mathsf{DCM}(1)$.
- There exists $L \in \mathsf{DCM}(1,1)$ (one-way), such that $\text{outf}^{-1}(L) \notin 2\mathsf{DCM}(1)$.
- There exists $L \in \mathsf{DCM}(1,1)$ (one-way), such that $\text{inf}^{-1}(L) \notin 2\mathsf{DCM}(1)$.
- There exists $L \in 2\mathsf{DCM}(1)$ with 1 input turn and 1 counter reversal, such that $\text{pref}^{-1}(L) \notin 2\mathsf{DCM}(1)$.
- There exists $L \in 2\mathsf{DCM}(1)$ with 1 input turn and 1 counter reversal, R regular, such that $LR \notin 2\mathsf{DCM}(1)$.

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

References

[1] Brenda S. Baker and Ronald V. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8(3):315–332, 1974.

[2] Ehsan Chiniforooshan, Mark Daley, Oscar H. Ibarra, Lila Kari, and Shinnosuke Seki. One-reversal counter machines and multihead automata: Revisited. *Theoretical Computer Science*, 454:81–87, 2012.

[3] Eitan M. Gurari and Oscar H. Ibarra. The complexity of decision problems for finite-turn multicounter machines. *Journal of Computer and System Sciences*, 22(2):220–229, 1981.

[4] YS Han and Derick Wood. The generalization of generalized automata: Expression automata. *International Journal of Foundations of Computer Science*, 16(03):499–510, 2005.

[5] M.A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley series in computer science. Addison-Wesley Pub. Co., 1978.

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

[7] Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, 1978.

[8] Oscar H. Ibarra, Tao Jiang, Nicholas Tran, and Hui Wang. New decidability results concerning two-way counter machines. *SIAM J. Comput.*, 23(1):123–137, 1995.

[9] H Jürgensen, L Kari, and G Thierrin. Morphisms preserving densities. *International Journal of Computer Mathematics*, 78:165–189, 2001.

[10] Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing Machines. *Annals of Mathematics*, 74(3):pp. 437–455, 1961.

[11] Cyril Nicaud. Average state complexity of operations on unary automata. In Mirosław Kutyłowski, Leszek Pacholski, and Tomasz Wierzbicki, editors, *Mathematical Foundations of Computer Science 1999*, volume 1672 of *Lecture Notes in Computer Science*, pages 231–240. Springer Berlin Heidelberg, 1999.