Deletion Operations on Deterministic Families of Automata

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. Many different deletion operations are investigated applied to languages accepted by one-way and two-way deterministic reversal-bounded multicounter machines as well as finite automata. Operations studied include the prefix, suffix, infix and outfix operations, as well as left and right quotient with languages from different families. It is often expected that language families defined from deterministic machines will not be closed under deletion operations. However, here, it is shown that one-way deterministic reversal-bounded multicounter languages are closed under right quotient with languages from many different language families; even those defined by nondeterministic machines such as the context-free languages, or languages accepted by nondeterministic pushdown machines augmented by any number of reversal-bounded counters. Also, it is shown that when starting with one-way deterministic machines with one counter that makes only one reversal, taking the left quotient with languages from many different language families, again including those defined by nondeterministic machines such as the context-free languages, yields only one-way deterministic reversal-bounded multicounter languages (by increasing the number of counters). However, if there are even just two more reversals on the counter, or a second 1-reversal-bounded counter, taking the left quotient (or even just the suffix operation) yields languages that can neither be accepted by deterministic reversal-bounded multicounter machines, nor by 2-way nondeterministic machines with one reversal-bounded counter. A number of other results with deletion operations are also shown.

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.

R. Jain et al. (Eds.): TAMC 2015, LNCS 9076, pp. 388–399, 2015.
DOI: 10.1007/978-3-319-17142-5_33

1 Introduction

This paper involves the study of various types of deletion operations applied to languages accepted by one-way deterministic reversal-bounded multicounter machines (DCM). These are machines that operate like finite automata with an additional fixed number of counters, where there is a bound on the number of times each counter switches between increasing and decreasing [2,10]. These languages have many decidable properties, such as emptiness, infiniteness, equivalence, inclusion, universe and disjointness [10].

These machines have been studied in a variety of different applications, such as to membrane computing, verification of infinite-state systems and Diophantine equations.

Recently, in [5], a related study was conducted for insertion operations; specifically operations defined by ideals obtained from the prefix, suffix, infix and outfix relations, as well as left and right concatenation with languages from different language families. It was found that languages accepted by one-way deterministic reversal-bounded counter machines with one reversal-bounded counter are closed under right concatenation with Σ^*, but having two 1-reversal-bounded counters and right concatenating Σ^* yields languages outside of DCM and 2DCM(1) (languages accepted by two-way deterministic machines with one counter that is reversal-bounded). It also follows from this analysis that the right input end-marker is necessary for even one-way deterministic reversal-bounded counter machines, when there are at least two counters. Also, concatenating Σ^* to the left of some one-way deterministic 1-reversal-bounded one counter languages yields languages that are neither in DCM nor 2DCM(1). Other recent results on reversal-bounded multicounter languages include a technique to show languages are outside of DCM [3].

Closure properties of some variants of nondeterministic counter machines under deletion operations were studied in [14]. However, in this paper we investigate deterministic machines which were not examined in [14].

2 Preliminaries

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

We assume knowledge of standard formal language theoretic concepts such as languages, finite automata, determinism, nondeterminism, semilinearity, recursive and recursively enumerable languages [2,9]. Next, we will give some notation used in the paper. The empty word is denoted by λ. If Σ is a finite alphabet, then Σ^* is 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 $a_i \in \Sigma$, $1 \le i \le n$, the length of w is denoted by $|w| = n$, and the reversal of w is denoted by $w^R = a_n \cdots a_1$. A language over Σ is any subset of Σ^*. Given a language $L \subseteq \Sigma^*$, the complement of L, $\Sigma^* \setminus L$ is denoted by $\overline{L}$. Given two languages L_1, L_2, the left quotient of L_2 by L_1, $L_1^{-1}L_2 = \{y \mid xy \in L_2, x \in L_1\}$, and the right quotient of L_1 by L_2 is $L_1 L_2^{-1} = \{x \mid xy \in L_1, y \in L_2\}$.

A language L is *word-bounded* or simply *bounded* if $L \subseteq w_1^* \cdots w_k^*$ for some $k \geq 1$ and (not-necessarily distinct) words $w_1, \ldots, w_k$. Further, L is *letter-bounded* if each w_i is a distinct letter. Also, L is *bounded-semilinear* if $L \subseteq w_1^* \cdots w_k^*$ and $Q = \{(i_1, \ldots, i_k) \mid w_1^{i_1} \cdots w_k^{i_k} \in L\}$ is a semilinear set [12].

We now present notation for common word and language operations used throughout the paper.

Definition 1. *For a language $L \subseteq \Sigma^*$, the prefix, suffix, infix and outfix operations are defined by:*

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

Note that $\text{pref}(L) = L(\Sigma^*)^{-1}$ and $\text{suff}(L) = (\Sigma^*)^{-1}L$.

The outfix operation has been generalized to the notion of embedding [13]:

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

Note that $\text{outf}(L) = \text{emb}(L, 1)$.

A *nondeterministic multicounter machine* is a finite automaton augmented by a fixed number of counters. The counters can be increased, decreased, tested for zero, or tested to see if the value is positive. A multicounter machine is *reversal-bounded* if every counter makes a fixed number of changes between increasing and decreasing.

Formally, 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 input end-marker, the initial state in Q, and the set of final states that is a subset of Q. The transition function δ (defined as in [10] except with only a right end-marker since we only use one-way inputs) is a mapping from $Q \times (\Sigma \cup \{\$\}) \times \{0,1\}^k$ into $Q \times \{\text{S}, \text{R}\} \times \{-1, 0, +1\}^k$, 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 direction of the input tape head movement is given by the symbols S are R for either *stay* or *right* respectively. The machine M is *deterministic* if δ is a function. A *configuration* of M is a $k+2$-tuple $(q, w\$, c_1, \ldots, c_k)$ for describing the situation where 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 $d \in \{\text{S}, \text{R}\}$ and $w' = aw$ if $d = \text{S}$, and $w' = w$ if $d = \text{R}$. Extended derivations are given by $\vdash_M^*$, the reflexive, transitive closure of $\vdash_M$. 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 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 $\mathsf{NCM} = \bigcup_{k,l\geq 0} \mathsf{NCM}(k,l)$ and $\mathsf{DCM} = \bigcup_{k,l\geq 0} \mathsf{DCM}(k,l)$. We will also sometimes refer to a multicounter machine as being in $\mathsf{NCM}(k,l)$ ($\mathsf{DCM}(k,l)$), if it has k l-reversal bounded counters (and is deterministic).

We denote by REG the family of regular languages, and by NPCM the family of languages accepted by nondeterministic pushdown automata augmented by a fixed number of reversal-bounded counters [10]. We also denote by $\mathsf{2DCM}(1)$ the family of languages accepted by two-way input, deterministic finite automata (both a left and right input tape end-marker are required) augmented by one reversal-bounded counter [11]. A machine of this form is said to be *finite-crossing* if there is a fixed c such that the number of times the boundary between any two adjacent input cells is crossed is at most c [6]. A machine is *finite-turn* if the input head makes at most k turns on the input, for some k. Also, 2NCM is the family of languages accepted by two-way nondeterministic machines with a fixed number of reversal-bounded counters, while 2DPCM is the family of two-way deterministic pushdown machines augmented by a fixed number of reversal-bounded counters.

The next result proved in [12] gives examples of weak and strong machines that are equivalent over word-bounded languages.

Theorem 1. *[12] The following are equivalent for every word-bounded language L:*

1. *L can be accepted by an* NCM.
2. *L can be accepted by an* NPCM.
3. *L can be accepted by a finite-crossing* 2NCM.
4. *L can be accepted by a* DCM.
5. *L can be accepted by a finite-turn* $\mathsf{2DCM}(1)$.
6. *L can be accepted by a finite-crossing* 2DPCM
7. *L is bounded-semilinear.*

We also need the following result in [11]:

Theorem 2. *[11] Let $L \subseteq a^*$ be accepted by a* 2NCM *(not necessarily finite-crossing). Then L is regular, hence, semilinear.*

3 Closure and Non-closure for Erasing Operations

3.1 Right Quotient for DCM

We begin by showing the closure of DCM under right quotient with any nondeterministic reversal bounded machine, even when augmented with a pushdown store.

Proposition 1. *Let $L_1 \in \mathsf{DCM}$ and let $L_2 \in \mathsf{NPCM}$. Then $L_1 {L_2}^{-1} \in \mathsf{DCM}$.*

Proof. Consider a DCM machine $M_1 = (k_1, Q_1, \Sigma, \$, \delta_1, s_0, F_1)$ and NPCM machine M_2 over Σ with k_2 counters where $L(M_1) = L_1$ and $L(M_2) = L_2$. A DCM machine M' will be constructed accepting $L_1 {L_2}^{-1}$.

Let $\Gamma = \{a_1, \dots, a_{k_1}\}$ be new symbols. For each $q \in Q_1$, let $M_c(q)$ be an interim $k_1 + k_2$ counter (plus a pushdown) NPCM machine over Γ constructed as follows: on input $a_1^{p_1} \cdots a_{k_1}^{p_{k_1}}$, $M_c(q)$ increments the first k_1 counters to $(p_1, \dots, p_{k_1})$. Then $M_c(q)$ nondeterministically guesses a word $x \in \Sigma^*$ and simulates M_1 on $x\$$ starting from state q and from the counter values of $(p_1, \dots, p_{k_1})$ using the first k_1 counters, while in parallel, simulating M_2 on x using the next k_2 counters and the pushdown. This is akin to the product automaton construction described in [10] showing NPCM is closed under intersection with NCM. Then $M_c(q)$ accepts if both M_1 and M_2 accept.

Claim. Let $L_c(q) = \{a_1^{p_1} \cdots a_{k_1}^{p_{k_1}} \mid \exists x \in L_2 \text{ such that } (q, x\$, p_1, \dots, p_{k_1}) \vdash_{M_1}^* (q_f, \$, p'_1, \dots p'_{k_1}), p'_i \geq 0, 1 \leq i \leq k_1, q_f \in F_1\}$. Then $L(M_c(q)) = L_c(q)$.

Proof. Consider $w = a_1^{p_1} \cdots a_{k_1}^{p_{k_1}} \in L_c(q)$. Then there exists x where $x \in L_2$ and $(q, x\$, p_1, \dots, p_{k_1}) \vdash_{M_1}^* (q_f^1, \$, p'_1, \dots p'_{k_1})$, where $q_f^1 \in F_1$. There must then be some final state $q_f^2 \in F_2$ reached when reading $x\$$ in M_2. Then, $M_c(q)$, on input w places $(p_1, \dots, p_{k_1}, 0, \dots, 0)$ on the counters and then can nondeterministically guess x letter by letter and simulate x in M_1 from state q on the first k_1 counters and simulate x in M_2 from its initial configuration on the remaining counters and pushdown. Then $M_c(q)$ ends up in state (q_f^1, q_f^2), which is final. Hence, $w \in L(M_c(q))$.

Consider $w = a^{p_1} \cdots a^{p_{k_1}} \in L(M_c(q))$. After adding each p_i to counter i, $M_c(q)$ guesses x and simulates M_1 on the first k_1 counters from q and simulates M_2 on the remaining counters from an initial configuration. It follows that $x \in L_2$, and $(q, x\$, p_1, \dots, p_{k_1}) \vdash_{M_1}^* (q_f^1, \$, p'_1, \dots p'_{k_1}), p'_i \geq 0, 1 \leq i \leq k_1, q_f^1 \in F_1$. Hence, $w \in L_c(q)$. ☐

Since for each $q \in Q_1$, $M_c(q)$ is in NPCM, it accepts a semilinear language [10], and since the accepted language is bounded, it is bounded-semilinear and can therefore be accepted by a DCM-machine by Theorem 1. Let $M'_c(q)$ be this DCM machine, with k' counters, for some k'.

Thus, a final DCM machine M' with $k_1 + k'$ counters is built as follows. In it, M' has k_1 counters used to simulate M_1, and also k' additional counters, used to simulate some $M'_c(q)$, for some $q \in Q_1$. Then, M' reads its input $x\$$, where $x \in \Sigma^*$, while simulating M_1 on the first k_1 counters, either failing, or reaching some configuration $(q, \$, p_1, \dots, p_{k_1})$, for some $q \in Q_1$, upon first hitting the end-marker $\$$. If it does not fail, we then simulate the DCM-machine $M'_c(q)$ on input $a_1^{p_1} \cdots a_{k_1}^{p_{k_1}}$, but this simulation is done deterministically by subtracting 1 from the first k_1 counters, in order, until each are zero instead of reading input characters, and accepts if $a_1^{p_1} \cdots a_{k_1}^{p_{k_1}} \in L(M'_c(q)) = L_c(q)$. Then M' is deterministic and accepts

$$\begin{aligned}
&\{x \mid \text{either } (s_0, x\$, 0, \ldots, 0) \vdash^*_{M_1} (q', a\$, p'_1, \ldots, p'_{k_1}) \vdash_{M_1} (q, \$, p_1, \ldots, p_{k_1}),\\
&\qquad a \in \Sigma, \text{ or } (s_0, x\$, 0, \ldots, 0) = (q, \$, p_1, \ldots, p_{k_1}), \text{ s.t. } a_1^{p_1} \cdots a_{k_1}^{p_{k_1}} \in L_c(q)\}\\
= &\{x \mid \text{either } (s_0, x\$, 0, \ldots, 0) \vdash^*_{M_1} (q', a\$, p'_1, \ldots, p'_{k_1}) \vdash_{M_1} (q, \$, p_1, \ldots, p_{k_1}),\\
&a \in \Sigma, \text{ or } (s_0, x\$, 0, \ldots, 0) = (q, \$, p_1, \ldots, p_{k_1}), \text{ where } \exists y \in L_2 \text{ s.t.}\\
&(q, y\$, p_1, \ldots, p_{k_1}) \vdash^*_{M_1} (q_f, \$, p''_1, \ldots, p''_{k_1}), q_f \in F_1\}\\
= &\{x \mid xy \in L_1, y \in L_2\}\\
= &L_1 L_2^{-1}.
\end{aligned}$$

□

These immediately show closure for the prefix operation.

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

We can modify this construction to show a strong closure result for one-counter languages that does not increase the number of counters.

Proposition 2. *Let* $l \in \mathbb{N}$. *If* $L_1 \in \mathsf{DCM}(1, l)$ *and* $L_2 \in \mathsf{NPCM}$, *then* $L_1 {L_2}^{-1} \in \mathsf{DCM}(1, l)$.

Proof. The construction is similar to the one in Proposition 1. However, we note that since the input machine for L_1 has only one counter, $L_c(q)$ is unary (regardless of the number of counters needed for L_2). Thus $L_c(q)$ is unary and semilinear, and Parikh's theorem states that all semilinear languages are letter-equivalent to regular languages [8], and all unary semilinear languages are regular. Thus $L_c(q)$ is regular, and can be accepted by a DFA.

We can then construct M' accepting $L_1 {L_2}^{-1}$ as in Proposition 1 without requiring any additional counters or counter reversals, by transitioning to the DFA accepting $L_c(q)$ when we reach the end of input at state q. □

Corollary 2. *Let* $l \in \mathbb{N}$. *If* $L \in \mathsf{DCM}(1, l)$, *then* $\text{pref}(L) \in \mathsf{DCM}(1, l)$.

In fact, this construction can be generalized from NPCM to any class of automata that can be defined using Definition 3. These classes of automata are described in more detail in [7]. We only define it in a way specific to our use in this paper. Only the first two conditions are required for Corollary 3, while the third is required for Corollary 5.

Definition 3. *A family of languages* $\mathscr{F}$ *is said to be* reversal-bounded counter augmentable *if*

– *every language in* $\mathscr{F}$ *is effectively semilinear,*
– *given* DCM *machine* M_1 *with* k *counters, state set* Q *and final state set* F, *and* $L_2 \in \mathscr{F}$, *we can effectively construct, for each* $q \in Q$, *the following language in* $\mathscr{F}$,

$$\{a_1^{p_1} \cdots a_k^{p_k} \mid \exists x \in L_2 \text{ such that } (q, x\$, p_1, \ldots, p_k) \vdash^*_{M_1} (q_f, \$, p'_1, \ldots p'_k),\\ p'_i \geq 0, q_f \in F\},$$

– *given* DCM *machine* M_1 *with* k *counters, state set* Q*, and* $L_2 \in \mathscr{F}$*, we can effectively construct, for each* $q \in Q$*, the following language in* $\mathscr{F}$*,*

$$\{a_1^{p_1} \cdots a_k^{p_k} \mid \exists x \in L_2 \text{ such that } (q, x, 0, \dots, 0) \vdash_{M_1}^{*} (q, \lambda, p_1, \dots p_k)\}.$$

There are many reversal-bounded counter augmentable families that L_2 could be from in this corollary, such as:

Corollary 3. *Let* $L_1 \in$ DCM *and* $L_2 \in \mathscr{F}$*, a family of languages that is reversal-bounded counter augmentable. Then* $L_1 {L_2}^{-1} \in$ DCM*. Furthermore, if* $L_1 \in$ DCM$(1, l)$ *for some* $l \in \mathbb{N}$*, then* $L_1 {L_2}^{-1} \in$ DCM$(1, l)$*.*

This construction could be applied to several other families of semilinear languages such as:

– MPCA's: one-way machines with k pushdowns where values may only be popped from the first non-empty stack, augmented by a fixed number of reversal-bounded counters [7].
– TCA's: NFA's augmented with a two-way read-write tape, where the number of times the read-write head crosses any tape cell is finitely bounded, again augmented by a fixed number of reversal-bounded counters [7].
– QCA's: NFA's augmented with a queue, where the number of alternations between the non-deletion phase and the non-insertion phase is bounded by a constant [7].
– EPDA's: embedded pushdown automata, modelled around a stack of stacks, introduced in [17]. These accept the languages of tree-adjoining grammars, a semilinear subset of the context-sensitive languages. As was stated in [7], we can augment this model with a fixed number of reversal-bounded counters and still get an effectively semilinear family.

3.2 Right and Left Quotients of Regular Sets

Let $\mathscr{F}$ be any family of languages (which need not be recursively enumerable). It is known that REG is closed under right quotient by languages in $\mathscr{F}$ [9]. However, this closure need not be effective, as it will depend on the properties of $\mathscr{F}$. The following is an interesting observation which connects decidability of the emptiness problem to effectiveness of closure under right quotient:

Proposition 3. *Let* $\mathscr{F}$ *be any family of languages which is effectively closed under intersection with regular sets and whose emptiness problem is decidable. Then* REG *is effectively closed under both left and right quotient by languages in* $\mathscr{F}$*.*

Proof. We will start with right quotient.

Let $L_1 \in$ REG and L_2 be in $\mathscr{F}$. Let M be a DFA accepting L_1. Let q be a state of M, and $L_q = \{y \mid M$ from initial state q accepts $y\}$. Let $Q' = \{q \mid q$ is a state of $M, L_q \cap L_2 \neq \emptyset\}$. Since $\mathscr{F}$ is effectively closed under intersection with regular sets and has a decidable emptiness problem, Q' is computable.

Then a DFA M' accepting $L_1 L_2^{-1}$ can be obtained by just making Q' the set of accepting states in M.

Next, for left quotient, let L_1 be in $\mathscr{F}$, and L_2 in REG be accepted by a DFA M whose initial state is q_0.

Let $L_q = \{x \mid M$ on input x ends in state $q\}$. Let $Q' = \{q \mid L_q \cap L_1 \neq \emptyset\}$. Then Q' is computable, since $\mathscr{F}$ is effectively closed under intersection with regular sets and has a decidable emptiness problem.

We then construct an NFA (with λ-transitions) M' to accept $L_1^{-1} L_2$ as follows: M' starting in state q_0 with input y nondeterministically goes to a state q in Q' without reading any input, and then simulates the DFA M. □

Corollary 4. REG *is effectively closed under left and right quotient by languages in:*

1. *the families of languages accepted by* NPCM *and* 2DCM(1) *machines,*
2. *the family of languages accepted MPCAs, TCAs, QCAs, and EPDAs,*
3. *the families of ET0L and Indexed languages.*

Proof. These families are closed under intersection with regular sets. They have also a decidable emptiness problem [1,7,16]. The family of ET0L languages and Indexed languages are discussed further in [16] and [1] respectively. □

3.3 Suffix, Infix and Left Quotient for DCM(1, 1)

In the case of one-counter machines that makes only one counter reversal, it will be shown that a DCM-machine that can accept their suffix and infix languages can always be constructed. However, in some cases, these resulting machines often require more than one counter. Thus, unlike prefix, DCM(1, 1) is not closed under suffix, left quotient, or infix. But, the result is in DCM.

The proof of Lemma 1 is quite lengthy, and due to space constraints is omitted but can be found online in [4]. We will give some intuition for the result here. First, DCM is closed under union and so the second statement of Lemma 1 follows from the first. For the first statement, an intermediate NPCM machine is constructed from L_1 and L that accepts a language L^c. This language contains words of the form qa^i where there exists some word w such that both $w \in L_1$, and also from the initial configuration of M (accepting L), it can read w and reach state q with i on the counter. Then, it is shown that this language is actually a regular language, using the fact that all semilinear unary languages are regular (as $(q)^{-1}L^c$ is unary; see [4] for full details). Then, DCM(1, 1) machines are created for every state q of M. These accept all words w such that $qa^i \in L^c$, and in M, from state q and counter i with w to read as input, M can reach a final state while emptying the counter. The fact that L^c is regular allows these machines to be created.

Lemma 1. *Let* $L \in \mathsf{DCM}(1,1), L_1 \in \mathsf{NPCM}$. *Then* $L_1^{-1} L$ *is the finite union of languages in* DCM(1, 1). *Furthermore, it is in* DCM.

From this, we obtain the following general result (proof also omitted due to space and is found in [4]).

Theorem 3. *Let* $L \in \mathsf{DCM}(1,1), L_1, L_2 \in \mathsf{NPCM}$*. Then both* $(L_1^{-1}L)L_2^{-1}$ *and* $L_1^{-1}(LL_2^{-1})$ *are a finite union of languages in* $\mathsf{DCM}(1,1)$*. Furthermore, both languages are in* DCM*.*

And, as with Corollary 3, this can be generalized to any language families that are reversal-bounded counter augmentable.

Corollary 5. *Let* $L \in \mathsf{DCM}(1,1), L_1 \in \mathscr{F}_1, L_2 \in \mathscr{F}_2$*, where* $\mathscr{F}_1$ *and* $\mathscr{F}_2$ *are any families of languages that are reversal-bounded counter augmentable. Then* $(L_1^{-1}L)L_2^{-1}$ *and* $L_1^{-1}(LL_2^{-1})$ *are both a finite union of languages in* $\mathsf{DCM}(1,1)$*. Furthermore, both languages are in* DCM*.*

As a special case, when using the fixed regular language Σ^* for the right and left quotient, we obtain:

Corollary 6. *Let* $L \in \mathsf{DCM}(1,1)$*. Then* $\mathrm{suff}(L)$ *and* $\mathrm{inf}(L)$ *are both* DCM *languages.*

It is however necessary that the number of counters increase to accept $\mathrm{suff}(L)$ and $\mathrm{inf}(L)$, for some $L \in \mathsf{DCM}(1,1)$. The result also holds for the outfix operator. The proof is omitted due to space and is found in [4].

Proposition 4. *There exists* $L \in \mathsf{DCM}(1,1)$ *where all of* $\mathrm{suff}(L), \mathrm{inf}(L), \mathrm{outf}(L)$ *are not in* $\mathsf{DCM}(1,1)$*.*

3.4 Non-closure of Suffix, Infix and Outfix with Multiple Counters or Reversals

In [5], a technique was used to show languages are not in DCM and $\mathsf{2DCM}(1)$ simultaneously. The technique uses undecidable properties to show non-closure. As $\mathsf{2DCM}(1)$ machines have two-way input and a reversal-bounded counter, it is difficult to derive “pumping” lemmas for these languages. Furthermore, unlike DCM and NCM machines, $\mathsf{2DCM}(1)$ machines 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 $\mathsf{2DCM}(1)$ whose counter makes only one reversal. However, $L_2 = \{a^i b^j c^k \mid i, j, k \geq 2, k = ij\}$ cannot be accepted by a $\mathsf{2DCM}(1)$ [11]. This technique from [5] works as follows. The proof uses the fact that there is a recursively enumerable but not recursive language $L_{\mathrm{re}} \subseteq \mathbb{N}_0$ that is accepted by a deterministic 2-counter machine [15]. 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 $n \in L_{\mathrm{re}}$).

Examining the constructions in [15] of the 2-counter machine demonstrates that the counters behave in a regular pattern. Initially one counter has some value d_1 and the other counter is zero. Then, 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 equals 0. During the phase, the positive counter decreases, while the other counter increases. The phase ends with the first counter containing 0 and the other counter containing d_{i+1}. In the next phase, the modes of the counters are interchanged. Thus, a sequence of configurations where the phases are changing 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. The second component of the configuration refers to the value of the first counter, and the third component refers to the value of the second. Also, notice 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 only occurs if the number is divisible by 2 or 3, respectively). The case applied is determined by q_i. Hence, a function h can be defined such that 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 one of $1, 2, 3, 1/2, 1/3$.

Let T be a 2-counter machine accepting a recursively enumerable language that is not recursive. 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 Q be the states of T, 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 for each i, $1 \leq i \leq 2m$, $I_i = q1^k$ 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 the first counter (resp., the second) 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);

Then, the following was shown in [5].

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

We will use this language exactly to show taking either the suffix, infix or outfix of a language in $\mathsf{DCM}(1, 3), \mathsf{DCM}(2, 1)$ or $\mathsf{2DCM}(1)$ can produce languages that are in neither DCM nor $\mathsf{2DCM}(1)$.

Theorem 4. *There exists a language L in all of $L \in \mathsf{DCM}(1, 3)$, $L \in \mathsf{DCM}(2, 1)$, and $L \in \mathsf{2DCM}(1)$ (which makes no turn on the input and 3 reversals on the counter) such that* $\mathrm{suff}(L) \notin \mathsf{DCM} \cup \mathsf{2DCM}(1)$, $\mathrm{inf}(L) \notin \mathsf{DCM} \cup \mathsf{2DCM}(1)$, *and* $\mathrm{outf}(L) \notin \mathsf{DCM} \cup \mathsf{2DCM}(1)$.

Proof. Let L_0 be the language defined above, which is not in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. Let a, b be new symbols. Clearly, bL_0b is also not in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. Let $L = \{a^i b\#I_1\#I_2\#\cdots\#I_{2m}\#b \mid I_1, \ldots, I_{2m}$ are configurations of the 2-counter machine T, $i \leq 2m-1, I_{i+1}$ is not a valid successor of $I_i\}$. Clearly L is in $\mathsf{DCM}(1,3)$, in $\mathsf{DCM}(2,1)$, and in $\mathsf{2DCM}(1)$ (as $\mathsf{DCM}(1,3)$ is a subset of $\mathsf{2DCM}(1)$).

Let L_1 be $\text{suff}(L)$. Suppose L_1 is in DCM (resp., $\mathsf{2DCM}(1)$). Then $L_2 = \overline{L_1}$ is also in DCM (resp., $\mathsf{2DCM}(1)$).

Let $R = \{b\#I_1\#I_2\cdots\#I_{2m}\#b \mid I_1,\ldots,I_{2m}$ are configurations of $T\}$. Then since R is regular, $L_3 = L_2 \cap R$ is in DCM (resp, $\mathsf{2DCM}(1)$). We get a contradiction, since $L_3 = bL_0b$.

Non-closure under infix and outfix can be shown similarly. □

This implies non-closure under left-quotient with regular languages, and this result also extends to the embedding operation, a generalization of outfix.

Corollary 7. *There exists* $L \in \mathsf{DCM}(1,3), L \in \mathsf{DCM}(2,1), L \in \mathsf{2DCM}(1)$ *(which makes no turn on the input and 3 reversals on the counter), and* $R \in \mathsf{REG}$ *such that* $R^{-1}L \notin \mathsf{DCM} \cup \mathsf{2DCM}(1)$.

Corollary 8. *Let* $m > 0$. *Then there exists* $L \in \mathsf{DCM}(1,3), L \in \mathsf{DCM}(2,1)$, $L \in \mathsf{2DCM}(1)$ *(which makes no turn on the input and 3 reversals on the counter) such that* $\text{emb}(L,m) \notin \mathsf{DCM} \cup \mathsf{2DCM}(1)$.

The results of Theorem 4 and Corollary 7 are optimal for suffix and infix as these operations applied to $\mathsf{DCM}(1,1)$ are always in DCM by Corollary 6 (and since $\mathsf{DCM}(1,2) = \mathsf{DCM}(1,1)$). But whether the outfix and embedding operations applied to $\mathsf{DCM}(1,1)$ languages is always in DCM is an open question.

3.5 Closure for Bounded Languages

In this subsection, deletion operations applied to bounded and letter-bounded languages will be examined.

The following is a required straightforward corollary to Theorem 2.

Corollary 9. *Let* $L \subseteq \#a^*\#$ *be accepted by a* $\mathsf{2NCM}$. *Then* L *is regular.*

Theorem 5. *If* L *is a bounded language accepted by either a finite-crossing* $\mathsf{2NCM}$*, an* NPCM *or a finite-crossing* $\mathsf{2DPCM}$*, then all of* $\text{pref}(L)$, $\text{suff}(L)$, $\inf(L)$, $\text{outf}(L)$ *can be accepted by a* DCM.

Proof. By Theorem 1, L can always be converted to an NCM. Further, one can construct NCM's accepting $\text{pref}(L), \text{suff}(L), \inf(L), \text{outf}(L)$ since one-way NCM is closed under prefix, suffix, infix and outfix. In addition, it is known that applying these operations on bounded languages produce only bounded languages. Thus, by another application of Theorem 1, the result can then be converted to a DCM. □

The "finite-crossing" requirement in the theorem above is necessary:

Proposition 5. *There exists a letter-bounded language* L *accepted by a* $\mathsf{2DCM}(1)$ *machine which makes only one reversal on the counter such that* $\text{suff}(L)$ *(resp.,* $\inf(L)$*,* $\text{outf}(L), \text{pref}(L)$*) is not in* $\mathsf{DCM} \cup \mathsf{2DCM}(1)$.

Proof. Let $L = \{a^i \# b^j \# \mid i, j \geq 2, j \text{ is divisible by } i\}$. Clearly, L can be accepted by a $\mathsf{2DCM}(1)$ which makes only one reversal on the counter. If $\text{suff}(L)$ is in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$, then $L' = \text{suff}(L) \cap \# b^+ \#$ would be in $\mathsf{DCM} \cup \mathsf{2DCM}(1)$. From Corollary 9, we get a contradiction, since L' is not semilinear. The other cases are shown similarly. □

References

1. Aho, A.V.: Indexed grammars–an extension of context-free grammars. J. ACM **15**(4), 647–671 (1968)
2. Baker, B.S., Book, R.V.: Reversal-bounded multipushdown machines. J. Comput. Syst. Sci. **8**(3), 315–332 (1974)
3. Chiniforooshan, E., Daley, M., Ibarra, O.H., Kari, L., Seki, S.: One-reversal counter machines and multihead automata: Revisited. Theor. Comput. Sci. **454**, 81–87 (2012)
4. Eremondi, J., Ibarra, O., McQuillan, I.: Deletion operations on deterministic families of automata. Technical report 2014–03, University of Saskatchewan (2014). http://www.cs.usask.ca/documents/techreports/2014/TR-2014-03.pdf
5. Eremondi, J., Ibarra, O.H., McQuillan, I.: Insertion operations on deterministic reversal-bounded counter machines. In: Dediu, A.-H., Formenti, E., Martín-Vide, C., Truthe, B. (eds.) LATA 2015. LNCS, vol. 8977, pp. 200–211. Springer, Heidelberg (2015)
6. Gurari, E.M., Ibarra, O.H.: The complexity of decision problems for finite-turn multicounter machines. J. Comput. Syst. Sci. **22**(2), 220–229 (1981)
7. Harju, T., Ibarra, O., Karhumäki, J., Salomaa, A.: Some decision problems concerning semilinearity and commutation. J. Comput. Syst. Sci. **65**(2), 278–294 (2002)
8. Harrison, M.: Introduction to Formal Language Theory. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company, Boston (1978)
9. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)
10. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM **25**(1), 116–133 (1978)
11. 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)
12. Ibarra, O.H., Seki, S.: Characterizations of bounded semilinear languages by one-way and two-way deterministic machines. Int. J. Found. Comput. Sci. **23**(6), 1291–1306 (2012)
13. Jürgensen, H., Kari, L., Thierrin, G.: Morphisms preserving densities. Int. J. Comput. Math. **78**, 165–189 (2001)
14. Kari, L., Seki, S.: Schema for parallel insertion and deletion: Revisited. Int. J. Found. Comput. Sci. **22**(07), 1655–1668 (2011)
15. Minsky, M.L.: Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. Ann. Math. **74**(3), 437–455 (1961)
16. Rozenberg, G., Salomaa, A.: The Mathematical Theory of L Systems. Academic Press Inc, New York (1980)
17. Vijayashanker, K.: A Study of Tree Adjoining Grammars. Ph.D. thesis, Philadelphia, PA, USA (1987)