The Effect of End-Markers on Counter Machines and Commutativity[☆]

Oscar H. Ibarra[a,1], Ian McQuillan[b,2]

[a] *Department of Computer Science*
University of California, Santa Barbara, CA 93106, USA
[b] *Department of Computer Science, University of Saskatchewan*
Saskatoon, SK S7N 5A9, Canada

Abstract

Restrictions of reversal-bounded multicounter machines are studied; in particular, those that cannot subtract from any counter until it has reached the end of the input. It is proven that this does not alter the languages accepted when the machines are nondeterministic. When the machines are deterministic, the languages (denoted by eDCM) are shown to coincide with those accepted by deterministic Parikh automata, but are strictly contained in the class of languages accepted by machines without this condition. It then follows that all commutative semilinear languages are in this restricted class. A number of decidability and complexity properties are shown, such as the ability to test, given a deterministic pushdown automaton (even if augmented by a fixed number of reversal-bounded counters), whether it is commutative. Lastly, this deterministic family, eDCM, is shown to be the smallest family of languages closed under commutative closure, right quotient with regular languages and inverse deterministic finite transductions.

Keywords: Counter Machines, Commutativity, Reversal-Bounds, Determinism, Finite Automata

1. Introduction

The *commutative closure* of a language L, comm(L), is the language of all words obtained by permuting the positions of the letters of all words in L. A language L is then *commutative* if comm(L) = L. The *Parikh map* of a word (and a language respectively), is the vector representing the number of copies of each letter in the word (the set of Parikh vectors of all words). These provide an equivalent criteria for commutative closure; comm(L) is the set of all words with the same Parikh vector as a word of L. Thus, studying the set of Parikh vectors is closely related to commutativity. It was found by Parikh [1] that every context-free language has a so-called *semilinear* (defined formally below in Section 2) Parikh map. The semilinear criteria can be equivalently expressed as, every language with a semilinear Parikh map has the same commutative closure as a regular language [2]. However, it is quite easy to create commutative languages that are not regular (nor context-free), such as $\{w \mid w$ has the same number of a's, b's and c's$\}$.

There is a model of automata that can accept every commutative semilinear language; namely the family of one-way nondeterministic reversal-bounded multicounter languages (NCM) [3, 4]. In [5], it was shown that NCM is in fact the smallest trio (closed under λ-free homomorphism, inverse homomorphism and intersection with regular languages) that is also closed under taking commutative closures. NCM is equal to the family of languages accepted by another model, Parikh automata [6]. It has also been shown that languages accepted by deterministic Parikh automata are closed under commutative closure.

[☆] DOI: https://doi.org/10.1016/j.tcs.2016.02.034

URL: ibarra@cs.ucsb.edu (Oscar H. Ibarra), mcquillan@cs.usask.ca (Ian McQuillan)

[1] Supported, in part, by NSF Grant CCF-1117708 (Oscar H. Ibarra).
[2] Supported, in part, by a grant from Natural Sciences and Engineering Research Council of Canada (Ian McQuillan).

Commutative semilinear languages (referred to as COM-SLIP in [7]) have also been studied. Since every semilinear language has the same commutative closure as a regular language, and the fact that all regular languages can be accepted by deterministic Parikh automata, it follows that COM-SLIP is contained inside the deterministic Parikh languages. The family of commutative semilinear languages has been extended to their closure under union and concatenation (referred to as COM-SLIP$^{\cdot,\cup}$ [7]), and these languages are also strictly contained in NCM, since NCM is closed under union and concatenation.

In this paper, one-way *deterministic* reversal-bounded multicounter languages (DCM) are studied, and a new restriction is introduced, eDCM, that are DCM machines that cannot subtract from any counter until hitting the end of the input. It is shown that this new family coincides with deterministic Parikh automata, and it therefore follows that both families are strictly contained inside DCM, and all commutative semilinear languages are contained in both. We then use the eDCM model to demonstrate a new language that can be accepted in DCM with only one counter that makes one counter reversal that is not in eDCM.

As these families are contained in the family of DCM languages, we explore a number of decidability and complexity properties that NCM does not have, such as decidable containment and equivalence problems. Several properties of commutative semilinear languages become easily decidable. For example, it is possible to test for either containment or equivalence between the commutative closures of any effectively semilinear languages (or between the commutative closure of any effectively semilinear language and an arbitrary DCM language). It is also shown that it is possible to decide whether an arbitrary DPCM language (a language accepted by a deterministic machine that has an unrestricted pushdown plus a fixed number of reversal-bounded counters) is commutative, and similarly for other deterministic automata models accepting semilinear languages. Also, testing membership in DCM is computable in logarithmic space on a deterministic Turing machine, and thus complexity theoretic results are presented for Turing Machines accepting semilinear languages. It is then shown that the concatenation closure of commutative semilinear languages is not always a DCM language, and therefore, the COM-SLIP$^{\cdot,\cup}$ languages are not contained in DCM; they are incomparable.

Finally, it is shown that eDCM (and hence deterministic Parikh automata) is the smallest family of languages closed under commutative closure, right quotient with regular languages, and inverse deterministic finite transductions. Such a characterization of a family of languages involving deterministic automata (that do not coincide with nondeterministic automata) using closure properties is somewhat unusual and is of interest.

2. Preliminaries

We assume familiarity with formal language and automata theory [8], and computational complexity theory [9]. We will fix the notation used in the paper. Let Σ be a finite alphabet. Then Σ^* (respectively Σ^+) is the set of all words (non-empty words) over Σ. A *word* is an element $w \in \Sigma^*$, λ is the *empty word*, and a *language* is any $L \subseteq \Sigma^*$. The *complement* of $L \subseteq \Sigma^*$ is $\overline{L} = \Sigma^* - L$. A language $L \subseteq \Sigma^*$ is *bounded* if there exists (not necessarily distinct) words $w_1, \ldots, w_k$ such that $L \subseteq w_1^* \cdots w_k^*$. Further, L is *letter-bounded* if there exists (not necessarily distinct) letters $a_1, \ldots, a_k$ such that $L \subseteq a_1^* \cdots a_k^*$. For $L, R \subseteq \Sigma^*$, the *right quotient* of L by R is $LR^{-1} = \{x \mid xy \in L, y \in R\}$.

Let $\mathbb{N}$ be the set of positive integers and $\mathbb{N}_0$ be the set of non-negative integers. Let $m \in \mathbb{N}_0$. Then $\pi(m)$ is 1 if $m > 0$ and 0 otherwise. Let $m \in \mathbb{N}$. Then $\mathbb{N}(m) = \{1, \ldots, m\}$. A subset Q of $\mathbb{N}_0^m$ is a *linear set* if there exist vectors $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_n \in \mathbb{N}_0^m$ such that $Q = \{\vec{v}_0 + i_1\vec{v}_1 + \cdots i_n\vec{v}_n \mid i_1, \ldots, i_n \in \mathbb{N}_0\}$. The vectors $\vec{v}_0$ (referred to as the *constant*) and $\vec{v}_1, \ldots, \vec{v}_n$ (referred to as *periods*) are called the *generators* of the linear set Q. A finite union of linear sets is called a *semilinear set*. Every finite subset of $\mathbb{N}_0^m$ (including the empty set $\emptyset$) is semilinear — it is just a finite union of linear sets with no periods. For semilinear sets $Q_1, Q_2 \subseteq \mathbb{N}^m$, $Q_1 + Q_2 = \{v \mid v = v_1 + v_2, v_1 \in Q_1, v_2 \in Q_2\}$.

Let $\Sigma = \{a_1, \ldots, a_m\}$. For a word w over Σ and a letter $a \in \Sigma$, we denote by $|w|_a$ the number of occurrences of a's in w, and by $|w|$ the length of w. The *Parikh map* of w is the m-dimensional vector $\psi(w) = (|w|_{a_1}, \ldots, |w|_{a_m})$. The *Parikh map* of a language $L \subseteq \Sigma^*$ is defined as $\psi(L) = \{\psi(w) \mid w \in L\}$. For $\vec{v} \in \mathbb{N}_0^m$, the inverse, $\psi^{-1}(\vec{v}) = \{w \in \Sigma^* \mid \psi(w) = \vec{v}\}$, extended to subsets of $\mathbb{N}_0^m$. A language is *semilinear*

if its Parikh map is a semilinear set. The *commutative closure* of $L \subseteq \Sigma^*$ is the set

$$\text{comm}(L) = \psi^{-1}(\psi(L)),$$

and a language L is said to be *commutative* if $L = \text{comm}(L)$.

Of interest are languages that are both commutative and semilinear. For example, $L_1 = \{w \mid w \in \{a,b\}^*, |w|_a = 2|w|_b\}$ and $L_2 = \{w \mid w \in \{a,b,c\}^*, 2|w|_a - 5|w|_b > 4|w|_c\}$ are commutative semilinear languages.

A *one-way k-counter machine* is denoted by $M = (k, Q, \Sigma, \triangleleft, \delta, q_0, F)$, where $Q, \Sigma, \triangleleft, q_0, F$ are the set of states, input alphabet, right input end-marker not in Σ, initial state in Q, and accepting states that are a subset of Q. The transition function δ is a relation from $Q \times (\Sigma \cup \{\triangleleft\}) \times \{0,1\}^k$ 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 enforce that the counters cannot store negative numbers. The symbols S and R indicate the direction that the input tape head moves, either *stay* or *right*. Further, M is *deterministic* if δ is a partial function. A configuration of M is a $k+2$-tuple $(q, w\triangleleft, c_1, \ldots, c_k)$ representing that M is in state q, with $w \in \Sigma^*$ still to read as input, and $c_1, \ldots, c_k \in \mathbb{N}_0$ being the contents of the k counters. The derivation relation $\vdash_M$ is defined between configurations, where $(q, aw, c_1, \ldots, c_k) \vdash_M (p, w', c_1 + d_1, \ldots, c_k + d_k)$ (we will sometimes also write $\vdash_M^t$ where t is a label associated with the transition t applied), 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 $d = \text{R}$. Let $\vdash_M^*$ be the reflexive, transitive closure of $\vdash_M$ (we will also write it as $\vdash_M^x$ where x is a word over labels associated with the transitions of M). A word $w \in \Sigma^*$ is accepted by M if $(q_0, w\triangleleft, 0, \ldots, 0) \vdash_M^* (q, \triangleleft, c_1, \ldots, c_k)$, 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. Furthermore, M is l-reversal-bounded if it operates in such a way that in every accepting computation, the count on each counter alternates between non-decreasing and non-increasing at most l times.

The families of $\mathsf{NCM}(k, l)$, for $k, l \geq 0$ is the family of one-way l-reversal-bounded k-counter languages, with $\mathsf{NCM} = \bigcup_{k,l \geq 0} \mathsf{NCM}(k, l)$. The family of context-free languages is denoted by NPDA. The family NPCM is the languages accepted by machines with one unrestricted pushdown, plus a fixed number of reversal-bounded counters. Nondeterministic Turing machines are denoted by NTM.

For each of the above, replacing N with D represents the deterministic variant. Machines with reversal-bounded counters have been extensively studied in the literature, see, e.g., [3, 4, 10].

The family of all commutative semilinear languages is denoted by $\mathsf{COM\text{-}SLIP}$, and the smallest family containing $\mathsf{COM\text{-}SLIP}$ closed under union and concatenation is $\mathsf{COM\text{-}SLIP}^{\cdot,\cup}$.

Next, Parikh automata will be defined [6, 11] and will be used for comparison with reversal-bounded counter machines. First, let Σ be an alphabet, $k \in \mathbb{N}$, and $C \subseteq \mathbb{N}_0^k$. The *projection on* Σ is the homomorphism θ from $(\Sigma \times C)^*$ to Σ^* that maps (a, d) to a, for all $a \in \Sigma, d \in C$. The *extended Parikh image*, $\overline{\psi}$, of $(a_1, d_1) \cdots (a_n, d_n) \in (\Sigma \times C)^*$ is $d_1 + \cdots + d_n$.

Definition 1. *Let Σ be an alphabet, $k \in \mathbb{N}$, and D a finite subset of $\mathbb{N}_0^k$. A Parikh automaton (PA) of dimension k is a pair (M, C), where $M = (Q, \Sigma \times D, \delta, q_0, F)$ is an NFA, and $C \subseteq \mathbb{N}_0^k$ is a semilinear set. The PA language,*

$$L(M, C) = \{\theta(w) \mid w \in L(M), \overline{\psi}(w) \in C\}.$$

The PA is deterministic, if for every state $q \in Q$, and every $a \in \Sigma$, there exists at most one pair $(p, d) \in Q \times D$ such that $p \in \delta(q, (a, d))$. The family of languages accepted by Parikh automata is NPA, and the family of languages accepted by deterministic Parikh automata is DPA.

3. eDCM and eNCM

The right input end-marker in the definition of DCM (and in other automata models) is of particular interest as it is often left off of one-way acceptor definitions. A different mode of acceptance, by *final state without end-marker*, was defined [12] that was unable to use its end-marker to detect the end of input. Languages accepted by DCM machines by final state without end-marker were called $\mathsf{DCM}_{\mathsf{NE}}$. With only one reversal-bounded counter, this did not change the languages accepted (so $\mathsf{DCM}(1, l) = \mathsf{DCM}_{\mathsf{NE}}(1, l)$,

for all l). But with two reversal-bounded counters that was not the case. There are indeed languages in $\mathsf{DCM}(2,1)$ that are not in $\mathsf{DCM}_{\mathsf{NE}}$ (with any number of counters). Therefore, the end-marker is necessary in general.

This is not the case with deterministic pushdown automata. If one defines language acceptance with these automata without an end-marker, then they are closed under right quotient with regular languages [13]. Thus, taking a language $L\$$, it is possible to remove the marker \$ with right quotient. In this case, the end-marker does not change the resulting language family.

With DCM however, the situation is different. DCM is indeed closed under right quotient with regular languages (also with NPCM languages) [14]. But the end-marker is *required* in order to prove closure under right quotient. If one takes an arbitrary DCM language L, it is true that $L\$$ is a $\mathsf{DCM}_{\mathsf{NE}}$ language. If (by contradiction), $\mathsf{DCM}_{\mathsf{NE}}$ were closed under right quotient with a single symbol, this would imply that L is in $\mathsf{DCM}_{\mathsf{NE}}$ as well, which would imply that $\mathsf{DCM} = \mathsf{DCM}_{\mathsf{NE}}$ as well, which is not the case. Therefore, we can conclude:

Proposition 2. $\mathsf{DCM}_{\mathsf{NE}}$ *is not closed under right quotient with single letters. Moreover,* DCM *is the smallest family of languages containing* $\mathsf{DCM}_{\mathsf{NE}}$ *that is closed under right quotient with single letters (or regular languages).*

The latter statement can be seen since it is possible to get every $L \in \mathsf{DCM}$ by taking $L\$ \in \mathsf{DCM}_{\mathsf{NE}}$ and taking the right quotient with \$, combined with the fact that DCM is closed under right quotient with regular languages.

This illustrates the importance of the end-marker, and its subtle influence on the languages that can be accepted for deterministic classes. In particular:

- Deterministic pushdown automata defined with or without an end-marker are identical. Both are closed under right quotient with regular languages [13].
- DCM (with end-marker) is strictly more powerful that $\mathsf{DCM}_{\mathsf{NE}}$ (without an end-marker) [12].
- DCM is closed under right quotient with regular languages (also for NPCM languages) [14].
- $\mathsf{DCM}_{\mathsf{NE}}$ is not closed under right quotient with a single symbol (Proposition 2).
- DCM is not closed under right concatenation with regular languages [12].
- $\mathsf{DCM}_{\mathsf{NE}}$ is closed under right concatenation with regular languages [12].

With the necessity of the end-marker established, an interesting question arises as to the families of languages that can be obtained by restricting the types of operations counter machines can apply when not scanning the end-marker. In this paper, we will restrict different classes of counter machines so that any instruction that reduces the size of the storage can only occur when the input tape is scanning the end-marker $\lhd$.

We define a simple restriction on NCM and DCM languages, called eNCM and eDCM. A 1-reversal-bounded k-counter NCM machine M is a k-counter eNCM machine if all decreasing transitions in M are defined on the right end-marker $\lhd$. Similarly with eDCM. Then let $\mathsf{eNCM}(k)$ be those languages (and machines) that are k-counter eNCM machines, and let $\mathsf{eNCM} = \bigcup_{k\geq 0} \mathsf{eNCM}(k)$, and similarly with $\mathsf{eDCM}(k)$ and eDCM. It is clear that for all k, $\mathsf{eDCM}(k) \subseteq \mathsf{DCM}(k,1), \mathsf{eNCM}(k) \subseteq \mathsf{NCM}(k,1)$, and $\mathsf{eDCM} \subseteq \mathsf{DCM}, \mathsf{eNCM} \subseteq \mathsf{NCM}$. It is known that each NCM and DCM language can be converted to a 1-reversal-bounded machine.

We will next compare NCM and eNCM.

Proposition 3. $\mathsf{NCM} = \mathsf{eNCM} = \mathsf{NPA}$.

Proof. It is already known that $\mathsf{NCM} = \mathsf{NPA}$ [6].

Hence, we need only show that an NCM can be converted to an equivalent eNCM. So, let $M \in \mathsf{NCM}(k,1)$ with k counters, $c_1, \ldots, c_k$. We may assume that M only accepts when it reaches the right end-marker and eventually enters an accepting state if and only if all its counters are zero.

We construct an eNCM M' with $2k$ counters, $c_1, d_1, \ldots, c_k, d_k$. On a given input w (with end-marker), M' simulates M faithfully using counters $c_1, \ldots, c_k$, as long as they are non-decreasing. If a counter c_i attempts to decrease, counter d_i is used to record the decrements (by adding 1 instead for every subtraction). At some point, M' guesses that the contents of d_i and c_i are equal (i.e., c_i would be zero if c_i was doing the decrementing). The simulation continues without using c_i and d_i using transitions defined on counter i being zero. If the simulated M accepts (by assumption, when this happens the input head of M is on the end-marker and all its counters are zero), M' then decrements all the counters and accepts if the contents of c_i and d_i become zero at the same time. □

Next, we will continue the study of eDCM and compare it to deterministic Parikh automata. To do this, we first need the following lemmas:

Lemma 4. *The following are known:*

1. *$C \subseteq \mathbb{N}_0^m$ is semilinear if and only if there is an NCM M_C accepting the language*

$$L_C = \{a_1^{k_1} \cdots a_m^{k_m} \mid (k_1, \ldots, k_m) \in C\}.$$

   Moreover, the conversion from the semilinear set to the NCM and vice-versa is effective [4].

2. *Any NCM accepting a bounded language, $L \subseteq w_1^* \cdots w_m^*$, can effectively be converted to an equivalent DCM [15].*

In fact, by using Proposition 3, the proof of item (2) above in [15] can be modified to show the following stronger result:

Proposition 5. *Any NCM accepting a bounded language, $L \subseteq w_1^* \cdots w_m^*$, where $w_1, \ldots, w_m \in \Sigma^*$, can effectively be converted to an equivalent eDCM.*

Thus, for any semilinear set $C \subseteq \mathbb{N}^m$, the language L_C in Lemma 4, Part (1) is an eDCM language, by Proposition 5.

Next, we compare eDCM to deterministic Parikh automata.

Lemma 6. eDCM $\subseteq$ DPA.

Proof. Let $M = (k, Q, \Sigma, \lhd, \delta, q_0, F)$ be an eDCM machine. Thus, M does not decrease any counter until hitting the end-marker $\lhd$. In M, we can assume without loss of generality that, for every $q \in Q, a \in \Sigma$ (not $\lhd$) and $x_1, \ldots, x_k, y_1, \ldots, y_k \in \{0, 1\}$, $\delta(q, a, x_1, \ldots, x_k) = \delta(q, a, y_1, \ldots, y_k)$, because M can keep track in the finite control of which counters are empty and which are non-empty since there is no subtraction until the end-marker. So, until the end-marker, the state and input letter completely determine the transition. Then, let $n = |Q|$. We can also assume without loss of generality that there are at most n consecutive stay transitions applied on Σ (this is not necessarily true on the end-marker) before a right transition as M can keep a counter in the state, and if there are at least $n+1$ stay transitions in a row, M is in an infinite loop, and can equivalently switch to a dead state.

For each $q \in Q$, consider the language $Y_q = \{a_1^{i_1} \cdots a_k^{i_k} \mid (q, \lhd, i_1, \ldots, i_k) \vdash_M^* (q_f, \lhd, j_1, \ldots, j_k), q_f \in F, j_1, \ldots, j_k \in \mathbb{N}_0\}$, where $a_1, \ldots, a_k$ are new symbols. This is a DCM language, by adding the input $(i_1, \ldots, i_k)$ to the counters, and then simulating M, and is therefore semilinear. Let C_q be the semilinear set such that $C_q = \psi(Y_q)$. Then it is immediate that $Y_q = L_{C_q}$, where L_{C_q} is from Lemma 4, Part (1). Thus, it is an eDCM language.

We are going to define a deterministic Parikh automaton (M_q, C_q), for an NFA M_q, that we will prove accepts the language

$$X_q = \{w \mid \begin{array}{l}(q_0, w\lhd, 0, \ldots, 0) \vdash_M^* (q, \lhd, i_1, \ldots, i_k) \vdash_M^* (q_f, \lhd, j_1, \ldots, j_k), q_f \in F \\ (q, \lhd, i_1, \ldots, i_k) \text{ is the first configuration in derivation to hit } \lhd\}.\end{array}$$

It is immediate that

$$X_q = \{w \mid \begin{array}{l}(q_0, w\triangleleft, 0, \ldots, 0) \vdash_M^* (q, \triangleleft, i_1, \ldots, i_k), a_1^{i_1} \cdots a_k^{i_k} \in Y_q \\ (q, \triangleleft, i_1, \ldots, i_k) \text{ is the first configuration in derivation to hit } \triangleleft\}.\end{array} \quad (1)$$

Indeed, it is clear that $L(M) = \bigcup_{q \in Q} X_q$. Further, since DPA is closed under union [6], it is sufficient to show that $L(M_q, C_q) = X_q$.

We now create the NFA M_q. For every $a \in \Sigma$, for every sequence of transitions of M, $\alpha : t_1 \cdots t_m$, where t_j is a label associated with transition $(p_j, T_j, l_1^j, \ldots, l_k^j) \in \delta(p_{j-1}, a, x_1^{j-1}, \ldots, x_k^{j-1})$, $T_j = \mathrm{S}$ for $1 \leq j < m$, $T_m = \mathrm{R}$ (thus implying $1 \leq m \leq n$ and therefore there are a finite number of these sequences, and they can be effectively determined), let $f(\alpha) = (l_1^1, \ldots, l_k^1) + \cdots + (l_1^m, \ldots, l_k^m)$. Then create the transition of M_q, from state p_0 to p_m on $(a, f(\alpha))$. The only final state of M_q is q. It is clear that (M_q, C_q) is deterministic. We will prove that $X_q = L(M_q, C_q)$.

"$\subseteq$" Let $w \in X_q$. Then

$$(q_0, w_0\triangleleft, l_1^0, \ldots, l_k^0) \vdash_M \cdots \vdash_M (q_x, w_x\triangleleft, l_1^x, \ldots, l_k^x),$$

$w_0 = w, l_1^0 = \cdots, l_k^0 = 0, q_x = q, w_x = \lambda, w_y \neq \lambda$ for all $y < x$, and $a_1^{l_1^x} \cdots a_k^{l_k^x} \in Y_q$. Let $t_1, \ldots, t_x$ be the labels associated with each transition in the derivation above. Let $j_1, \ldots, j_p$ be exactly those numbers such that $1 \leq j_1 < \cdots < j_p = x$ and $t_{j_1}, \ldots, t_{j_p}$ are transitions that read an input letter, and let $j_0 = 0$. Then $\alpha_r = t_{j_{r-1}+1} \cdots t_{j_r}, 1 \leq r \leq p$. Let $b_r \in \Sigma$ be such that all transitions in α_r are defined on b_r, $1 \leq r \leq p$. Then, there is a transition of M_q from q_0 to q_{j_1} on $(b_1, f(\alpha_1)), \ldots$, from $q_{j_{p-1}+1}$ to q_{j_p} on $(b_p, f(\alpha_p))$. Further, $f(\alpha_1) + \cdots + f(\alpha_p) = (l_1^x, \ldots, l_k^x) \in C_q$. Thus, $w \in L(M_q, C_q)$.

"$\supseteq$" Let $q_0, q_i, b_i, (l_1^i, \ldots, l_k^i), 1 \leq i \leq p$ be such that there is a transition from q_{i-1} to q_i on $(b_i, (l_1^i, \ldots, l_k^i))$, for all i, $1 \leq i \leq p$, $w = b_1 \cdots b_p \in L(M_q), q_p = q, \sum_{1 \leq i \leq p} (l_1^i, \ldots, l_k^i) \in C_q$. Then, by construction, for each i, there is a sequence of transitions α_i in M, all on b_i, where all but the last are stay transitions, and the last is a right transition, starting at q_{i-1} and ending in q_i and adding $(l_1^i, \ldots, l_k^i)$. Thus,

$$(q_0, b_1 \cdots b_p\triangleleft, 0, \ldots, 0) \vdash_M^* (q_1, b_2 \cdots b_p\triangleleft, l_1^1, \ldots l_k^1) \vdash_M^* \cdots \vdash_M^* (q_p = q, \triangleleft, \sum_{1 \leq i \leq p} l_1^i, \ldots, \sum_{1 \leq i \leq p} l_k^i),$$

and $(\sum_{1 \leq i \leq p} l_1^i, \ldots, \sum_{1 \leq i \leq p} l_k^i) = \sum_{1 \leq i \leq p} (l_1^i, \ldots, l_k^i) \in C_q$. Hence, from the last configuration of this derivation, then M can reach final state, by the construction of C_q. □

Proposition 7. eDCM = DPA.

Proof. By Lemma 6, it is sufficient to show that DPA $\subseteq$ eDCM.

Let $k \in \mathbb{N}$, and let (M, C) be a deterministic Parikh automaton of dimension k, where $M = (Q, \Sigma \times D, \delta, q_0, F)$. Then, for every transition $t : p \in \delta(q, (a, d))$, let $\max_t = \max_{1 \leq j \leq k} \{i_j \mid d = (i_1, \ldots, i_k)\}$.

For every semilinear set C, consider the language L_C from Lemma 4, Part (1). This is in eDCM. Then we construct an eDCM M' from DPA (A, C) as follows. M' simulates each transition $t : p \in \delta(q, (a, d))$ as follows: M' makes $\max_t$ transitions on a, all but the last being stay transitions, and the last being a right transition which switches from state q to p, and adds d to the counters (all using additions by 1 or 0 only in each counter which is clearly possible). And, on $\triangleleft$, M' verifies that it is in a final state of M and that the counter contents are in C by simulating M_C on the counter contents by subtracting one for every non-empty counter, from 1 to k, for every letter read in the simulation (simulating M_C can require additional counters). Indeed, this can be done deterministically since $L_C \in$ eDCM. Hence, $L(M') = L(M, C)$. □

From this, many known results regarding DPA apply to eDCM as well. For example, it is known that DPA $\subsetneq$ DCM, and so eDCM $\subsetneq$ DCM follows as well. However, the example used to separate DCM and DPA in [6] needs two counters (as it is not context-free). So, we will separate them in a stronger result with a language that only needs one counter that makes one reversal and is therefore context-free.

Let $v \in \Sigma^*$. Then define $\mathrm{suff}_i(v)$ to be the suffix of v of length i, if it exists, and undefined otherwise.

Proposition 8. eDCM = DPA $\subsetneq$ DCM. *Moreover, there exists* $L \in \mathsf{DCM}(1,1)$ *such that* $L \notin$ eDCM = DPA.

Proof. Consider the language $L = \{c^i v \mid v \in \{a,b\}^*, \mathrm{suff}_{|v|-i}(v) \in a^*\} \subseteq \{a,b,c\}^*$. Then $L \in \mathsf{DCM}(1,1)$ by counting the number of c's, and for every character of v until the counter is empty, reduce the counter by 1. Then when the counter reaches 0, there are $|v| - i$ characters left in the input. Verify that all remaining characters are a.

Assume that $L \in$ eDCM. Then then exists $M = (k, Q, \Sigma, \triangleleft, \delta, q_0, F)$ be an eDCM machine accepting L. Assume without loss of generality that all k counters increase immediately at the start of the computation and that $\delta(q, a, x_1, \ldots, x_k) = \delta(q, a, y_1, \ldots, y_k)$, for every $q \in Q, a \in \Sigma, x_1, \ldots, x_k, y_1, \ldots, y_k \in \{0,1\}$ (as in the proof of Lemma 6). When reading the section of c's, the machine M must operate similarly to a unary DFA whose structure is relatively simple. Unary DFAs have a "tail", a sequence of less then $|Q|$ states, whereby there is a transition from each state to the next in the sequence without repeats, followed by a "loop", a sequence of at most $|Q|$ states whereby there is a transition from each state to the next state in the sequence, and the last back to the first (and no other transitions) [16]. This is true for eDCM over one letter alphabets as well (although DCM has 'stay' transitions that do not exist for DFAs). Let t be the number of states in the tail, and let p be the number of states in the loop. Let j be the jth state of the loop, for j, from 0 to $p-1$ (where state 0 is the state where the tail connects to the loop).

Let $\gamma_j(y), 0 \leq j < p, y \in \{a,b\}^*$ be the vector $(q', c_1, \ldots, c_k)$, where from state j, reading y takes M to state q' and increases counter i by c_i. Thus, $(j, y, 0, \ldots, 0) \vdash_M^* (q', \lambda, c_1, \ldots, c_k)$, and therefore the final transition of this derivation must not be a stay transition. Note that $0, \ldots, 0$ can be replaced by any other counter values where this derivation increases the counter contents by $(c_1, \ldots, c_k)$.

Let j be such that $0 \leq j < p$. Let $m = |Q| + 1$. Let $u = b^m a^m$. Then there exists l, r such that $x = u^l (u)^r (u)^r$, and M traverses the same sequence of states and transitions in the last two sections of $(u)^r$ (eventually, M must hit the same state when beginning two sections of $b^m a^m$ since there are only $|Q|$ states, and thus repeating the characters between those two states will repeat the sequence of states and transitions traversed). Then within the section of a's in the last two sections of $(b^m a^m)^r$, there must exist $q \in Q$ such that M hits state q after reading a^α, and again after a^β, $\alpha < \beta$ by the pigeonhole principle, and since $m = |Q| + 1$. Then consider $y = u^l (b^m a^m)^{r-1} b^m a^{m-(\beta-\alpha)} (b^m a^m)^{r-1} b^m a^{m+(\beta-\alpha)}$. Then $\gamma_j(x) = \gamma_j(y)$ because the final state is the same by moving over the section between states q and itself from one to the other, and the number of applications of each transition is the same. However, if $i = |y| - (m + \beta - \alpha)$, then $c^i y \in L$ since the last $m + \beta - \alpha$ characters are a's. Thus, $c^i x \in L$, since $\gamma_j(x) = \gamma_j(y)$, a contradiction, since the last $m + \beta - \alpha$ characters of x are not a. □

The following known result on deterministic Parikh automata [6] now follows for eDCM:

Proposition 9. DPA = eDCM *is closed under commutative closure, but not under concatenation.*

This implies that every commutative semilinear language is in eDCM, since the commutative closure of every semilinear language is equal to the commutative closure of a regular language. Clearly, not all eDCM languages are commutative and so the inclusion is strict.

Proposition 10. COM-SLIP $\subsetneq$ DPA = eDCM.

This also demonstrates the following:

Proposition 11. *The following statements are equivalent for every commutative language* L*:*

1. L *is the commutative closure of a regular language.*
2. L *is the commutative closure of a context-free language.*
3. L *is the commutative closure of an* NPCM *language.*
4. L *is the commutative closure of a semilinear language.*
5. L *is in* eDCM = DPA.

6. L is in DCM.

Note, this applies for arbitrary Turing machines also accepting L that are commutative (not necessarily constructively, but it is constructive for the families of regular, context-free, and NPCM languages).

This also shows that for an NPCM language L, $\text{comm}(L)$ is in a smaller language family (eDCM) than L itself, and is even deterministic.

Lemma 12. *The concatenation closure of* COM-SLIP *languages is not always in* DCM.

PROOF. Assume otherwise. In Theorem 13 of [12], the language $L = \{w \mid w \in \{a, b, \#\}^*, |w|_a \neq |w|_b\}$ is considered over the alphabet $\Sigma = \{a, b, \#\}$. It is proven that $\#L\#\Sigma^* \notin$ DCM. But $\{\#\}, L, \Sigma^*$ are all commutative and in DCM, and hence semilinear, and therefore $\#L\#\Sigma^* \in$ DCM by the assumption, a contradiction. □

Proposition 13. COM-SLIP$^{\cdot,\cup}$ *and* DCM *(*eDCM, DPA *respectively) are incomparable.*

PROOF. The fact that COM-SLIP$^{\cdot,\cup} \not\subseteq$ DCM follows from Lemma 12. The other direction follows from the fact that all COM-SLIP languages on a two letter alphabet are context-free [7], therefore their union and concatenation are also context-free. But an eDCM can accept the non-context-free language $\{a^n b^n a^n \mid n > 0\}$. □

Hence, COM-SLIP $\subsetneq$ eDCM, and COM-SLIP$^{\cdot,\cup} \not\subseteq$ DCM, but COM-SLIP$^{\cdot,\cup} \subsetneq$ NCM ([7], where the equivalent BLIND counter formulation is used instead of NCM).

Since all commutative semilinear languages are in DCM, this provides some benefits due to improved decidability and complexity results. It is known that DCM has a decidable membership, emptiness, infiniteness, disjointness, containment and equivalence problems [4].

Proposition 14. *Let $\mathcal{L}$ be a language family that is effectively semilinear (such as* NPCM*). Then membership, emptiness, infiniteness, disjointness, containment and equivalence are decidable for commutative closures of languages in $\mathcal{L}$.*

For this, the languages need only be converted to languages in DCM, and then decision problems are obtained using DCM machines. Even stronger, containment, disjointness, equivalence, etc. can be decided when one language is the commutative closure of a language from $\mathcal{L}$ and the other language is a DCM language.

Proposition 15. *The following problems are decidable:*

1. *Given an* NPCM M_1 *and a* DPCM M_2*, is* $\text{comm}(L(M_1)) \subseteq L(M_2)$*?*
2. *Given* NPCM*s* M_1 *and* M_2*, is* $L(M_1) \subseteq \text{comm}(L(M_2))$*?*
3. *Given* DPCM M_1 *and* NPCM M_2*, is* $L(M_1) = \text{comm}(L(M_2))$*?*

PROOF. For Part 1, from Proposition 11, we can construct a DCM M_1' accepting $\text{comm}(L(M_1))$. Then, we can construct a DPCM M accepting $L(M_1') \cap \overline{L(M_2)}$ (DPCM is closed under complement and DPCM is closed under intersection with DCM [4]). The result follows since $\text{comm}\, L(M_1) \subseteq L(M_2)$ if and only if $L(M) = \emptyset$, and emptiness for NPCMs is decidable.

For Part 2, we construct a DCM M_2' accepting $\text{comm}(L(M_2))$. Then we construct a NPCM M accepting $L(M_1) \cap \overline{L(M_2')}$. Clearly, $L(M_1) \subseteq \text{comm}(L(M_2))$ if and only if $L(M) = \emptyset$, which is decidable.

Clearly, Part 3 follows from Parts 1 and 2. □

Proposition 15 can be generalized to other families of semilinear languages listed in [17], where NPCM can be replaced with nondeterministic versions of the machines, and DPCM can be replaced with the deterministic versions closed under complement. Examples of such machines include the nondeterministic and deterministic versions of arbitrary Turing machines, with a one-way read-only input, and two-way read-/write worktape that is finite crossing, i.e. the number of times the head crosses the boundary between two adjacent cells is bounded by a constant across all computations.

By Part 3, and by letting $M_1 = M_2 = M$:

Corollary 16. *It is decidable, given a* DPCM M*, whether the language* $L(M)$ *is commutative.*

However, for nondeterministic families:

Proposition 17. *It is undecidable, given an* $\mathsf{NCM}(1,1)$ M*, whether* $L(M)$ *is commutative.*

Proof. We reduce the problem to the undecidability of the halting problem for DTMs on an initially blank tape. Let Z be a DTM. We may assume that if Z halts, its halting sequence of computation is unique. Construct an $\mathsf{NCM}(1,1)$ M that accepts all words that are not a sequence of IDs (configurations) — a regular language — together with those that are a sequence of configurations but not a halting sequence of IDs of Z. This is a method used for example in [3] to show that the language of all invalid computations of a DTM can be accepted by an $\mathsf{NCM}(1,1)$ machine. Intuitively, all sequences of IDs that are not a halting sequence of Z can be accepted by nondeterministically guessing two consecutive configurations in the sequence, and verifying that the second does not follow from the first. Indeed, by scanning the state and read/write head position of both configurations, then M can either determine that their lengths imply that the second does not follow from the first, or otherwise, there is some position (nondeterministically guessed) of the first configuration that together with this position of the second configuration imply that the second configuration does not follow. Let Σ be the alphabet used in representing the sequences of IDs. Clearly, $L(M) = \Sigma^*$ which is commutative if Z does not halt, and $L(M) = \Sigma^* - \{x\}$ (where x is unique and represents the halting sequence of IDs of Z) is not commutative if Z halts. Hence, $L(M)$ is commutative if and only if $L(M) = \Sigma^*$ if and only if Z does not halt, which is undecidable. □

Next, we discuss some results on commutative closure with respect to Turing machines. We note the following regarding semilinear languages L, and recognizing $\mathrm{comm}(L)$ with a deterministic Turing machine in $\log n$ space.

Proposition 18. *Let* L *be any language whose Parikh map is semilinear. Then* $\mathrm{comm}(L)$ *can be accepted by a* DCM*; hence, also by a one-way* $\log n$ *space-bounded* DTM*. Further, if* $\mathrm{comm}(L)$ *is not a regular language, then it cannot be accepted in less than* $\log n$ *space.*

Proof. The first part follows from Proposition 9 and Proposition 11. For the second part, it is known that any DCM operates in linear time [3]; hence the numbers stored in its reversal-bounded counters are linear in the length of the input. It follows that $\mathrm{comm}(L)$ can be accepted by a $\log n$ space-bounded DTM. The last statement follows since it is known that any one-way $S(n)$ space-bounded NTM where $S(n)$ grows slower than $\log\ n$ accepts only regular languages [18]. Thus, if $\mathrm{comm}(L)$ is not regular, it requires at least $\log n$ space. □

For arbitrary Turing machines (not necessarily accepting semilinear languages), the following is true:

Proposition 19. *If* L *is a language accepted by a one-way (read-only) input* NTM *that is* $S(n)$ *space-bounded where* $S(n) \geq \log n$*, then* $\mathrm{comm}(L)$ *can also be accepted by a one-way* $S(n)$ *space-bounded* NTM*. If* $S(n) < \log n$*, then* L *is a regular language, and in this case, if* $\mathrm{comm}(L)$ *is not a regular language, then it cannot be accepted in less than* $\log n$ *space.*

Proof. Given a language L accepted by a one-way $S(n)$ space-bounded NTM M over input alphabet $\Sigma = \{a_1, \ldots, a_k\}$, construct a one-way $S(n)$ space-bounded NTM M', which operates as follows, when given input w:

1. M' scans the input and stores the values $|w|_{a_1}, \ldots, |w|_{a_k}$ in k counters $c_1, \ldots, c_k$, using log space.
2. M' then guesses some string x (symbol-by-symbol) and simulates M on x, decrementing counter c_i if it guesses that the next symbol is a_i. M' accepts if and only if the counters are all zero when M accepts. Clearly, M is $S(n)$ space-bounded and accepts $\mathrm{comm}(L)$.

For the second part, it is known that any one-way $S(n)$ space-bounded NTM where $S(n)$ grows slower than $\log\ n$ accepts only regular languages [18]. Therefore, if $S(n) < \log n$, then L must be regular, hence semilinear. Thus, Proposition 18 must apply and the statement follows. □

For bounded languages, we have the following result for deterministic Turing machines:

Proposition 20. *Let $S(n) \geq \log n$. If $L \subseteq w_1^* \cdots w_k^*$ (where $w_1, \ldots, w_k$ are not-necessarily distinct, fixed, non-null words) is accepted by a one-way or two-way $S(n)$ space-bounded* DTM *M, then* comm(L) *can also be accepted by a one-way $S(n)$ space-bounded* DTM *M'.*

Proof. Let $\Sigma = \{b_1, \ldots, b_l\}$ be the the input alphabet of M, and let $w \in \Sigma^*, |w| = n$. Then $w \in \text{comm}(L)$ if and only if there exists $v \in \text{comm}(w)$ such that $v \in w_1^* \cdots w_k^* \cap L$. Further, $w \in \text{comm}(L)$ if and only if there exists $v \in \text{comm}(w)$ such that $v \in \{w_1^{x_1} \cdots w_k^{x_k}\} \cap L$ for some $x_1, \ldots, x_k$, where each $x_i \leq n$ (and therefore each x_i requires at most $\lceil \log_2 n + 1 \rceil$ bits). Then we construct M' as follows: M' uses a counter (in binary) for each of $w_1, \ldots, w_k$. Call these counters $c_1, \ldots, c_k$. M' also uses l counters $d_1, \ldots, d_l$ and an additional l counters $e_1, \ldots, e_l$, and a single counter f.

To start, let $w = a_1 a_2 \cdots a_n, n \geq 0$ be the input, $a_i \in \Sigma$, for $1 \leq i \leq n$. M' records $|w|_{b_j}$ in tape e_j, for all j, $1 \leq j \leq l$, and the number n in counter f in binary. Then, on tapes $c_1, \ldots, c_k$, M' writes the numbers $i_1, \ldots, i_k$, for all possible values of i_j between 0 and n, one combination at a time. Each counter combination can be calculated deterministically within the space bounds by first placing n (the contents of f) within each tape $c_1, \ldots, c_k$ then subtracting one from the last non-empty counter c_i and setting each of $c_{i+1}, \ldots, c_k$ to n, until all counters are zero. For each of these counter combinations, M' then stores the number (in binary) of b_j's in $w_1^{i_1} \cdots w_k^{i_k}$ in counter d_j, by adding the contents of tape c_i to tape d_j, $|w_i|_{b_j}$ times, for all $i, j, 1 \leq i \leq k, 1 \leq j \leq l$.

Next, M' checks if all values in each d_j, $1 \leq j \leq l$, is equal to the value in the counter e_j, for all j, $1 \leq j \leq l$. If all are true, then $w_1^{i_1} \cdots w_k^{i_k}$ is a permutation of the input. In this case, M' then checks if $w_1^{i_1} \cdots w_k^{i_k}$ is in $L(M)$. If this is true for at least one permutation, M' accepts. □

It is still an open problem as to whether there exists an $S(n) \geq \log n$ space-bounded DTM M, whereby comm$(L(M))$ cannot be accepted by an $S(n)$ space-bounded DTM. But from Propositions 20 and 18, if such an M exists, then $S(n)$ must be less than n, $L(M)$ must be non-bounded, and it must not be semilinear. We conjecture that there does exist such a DTM. However, we remark that there are non-semilinear, non-bounded languages, such as $L = \{\#a^1\#a^2\#\cdots\#a^n | n > 0\}$, whereby L and comm(L) can be accepted by one-way $\log n$ space-bounded DTMs. Indeed, L is not bounded and its Parikh map is $\{(n(n+1)/2, n) \mid n > 0\}$ is not semilinear. For cases where $S(n) < \log n$, $L(M)$ must be a regular language and therefore requires no space, but if comm(L) is not regular, then it requires at least $\log n$ space. And since there are regular languages L such that comm(L) is not regular, comm(L) does require more space than L.

4. Characterization of eDCM

In [5], it was shown that NCM is the smallest family of languages that is a trio closed under commutative closure. We know from Proposition 11 that eDCM is closed under commutative closure also. Further, eDCM is closed under intersection with regular languages, inverse homomorphism, but not homomorphism [14]. In this section, it is shown that eDCM is the smallest family of languages (containing $\{\lambda\}$) that is closed under commutative closure, inverse deterministic finite transductions, and right quotient with regular languages.

The transducers are defined slightly differently than usual, to have a right input end-marker (note that a DCM has also a right end-marker), and to move right or stay on the input (like DCM, corresponding to transitions on a letter or the empty word). We will then start by showing that DCM and eDCM are closed under inverse deterministic versions of these transductions. In fact, these transducers can also be defined to also have a fixed number of reversal-bounded counters, and it is shown in [19] that DCM is closed under inverse deterministic transducers augmented by counters. However, this result does not hold for eDCM, and so we will provide a proof in this paper for only inverse transducers without counters.

A finite transducer is a tuple $A = (Q, \Sigma, \Gamma, \lhd, \delta, q_0, F)$ where $Q, \Sigma, \Gamma, \lhd, q_0, F$ are respectively the sets of states, input alphabet, output alphabet, right end-marker (not in $\Sigma \cup \Gamma$), initial state $q_0 \in Q$, and set of final states $F \subseteq Q$. The transition function is a finite relation from $Q \times (\Sigma \cup \{\lhd\})$ into $Q \times \{\text{R}, \text{S}\} \times \Gamma^*$. M is deterministic if δ is a partial function and if $\delta(F \times \{\lhd\}) = \emptyset$ to prevent multiple outputs from the same input

on deterministic transducers. A configuration of A is of the form $(q, w\triangleleft, z)$, where $q \in Q$ is the current state, $w \in \Sigma^*$ is the remaining input, and $z \in \Gamma^*$ is the accumulated output. Then, $(q, aw, z) \vdash_A (p, w', z'), a \in \Sigma \cup \{\triangleleft\}, aw, w' \in \Sigma^*\triangleleft$, where $(p, d, x) \in \delta(q, a), z' = zx, (d = \mathrm{S} \Rightarrow aw = w')$, and $(d = \mathrm{R} \Rightarrow w = w')$. Then $\vdash_A^*$ is the reflexive-transitive closure of $\vdash_A$.

Let $A = (Q, \Sigma, \Gamma, \triangleleft, \delta, q_0, F)$ be a finite transducer. For $L \subseteq \Sigma^*$, then $A(L) = \{x \mid (q_0, w\triangleleft, \lambda) \vdash_A^* (q_f, \triangleleft, x), w \in L, q_f \in F\}$. For $L \subseteq \Gamma^*$, then $A^{-1}(L) = \{w \mid (q_0, w\triangleleft, \lambda) \vdash_A^* (q_f, \triangleleft, x), x \in L, q_f \in F\}$.

Lemma 21. DCM *and* eDCM *are closed under inverse deterministic finite transductions.*

Proof. Let $M = (k, Q, \Gamma, \triangleleft, \delta, q_0, F)$ be a k-counter 1-reversal-bounded DCM (without loss of generality). Let $A = (Q_A, \Sigma, \Gamma, \triangleleft, \delta_A, q_A, F_A)$ be a deterministic finite transducer.

Then we construct a DCM machine $M' = (k, Q', \Sigma, \triangleleft, \delta', q_0', F')$ accepting $A^{-1}(L(M))$ as follows: M' takes as input a word $a_1 \cdots a_n \in \Sigma^*, a_i \in \Sigma, 1 \leq i \leq n$ followed by the end-marker $\triangleleft$. In the states of Q', M' keeps a buffer of at most length $\alpha = \max\{|x| \mid (p, d, x) \in \delta_A(q, a)\} + 1$. Then on each letter, a_i, M' simulates one transition of A on a_i, and stores the (deterministically calculated) output in the buffer. If the buffer becomes non-empty, M' simulates M on the buffer and the k counters. Once the buffer becomes empty again, M' continues the simulation of A (on a_i if the transition of A applied last was a stay transition, and on a_{i+1} if it was a right transition). If M' reaches the end-marker of A, and A is in a final state, then M' puts the end-marker $\triangleleft$ at the end of the output buffer. If this occurs, then M' continues simulating M on the buffer, accepting if it reaches a state of F with only $\triangleleft$ in the buffer. Hence, $L(M') = \{w \mid (q_A, w\triangleleft, \lambda) \vdash_A^* (q_f, \triangleleft, x), x \in L(M), q_f \in F_A\}\} = A^{-1}(L(M))$. Also, M' is deterministic since the output buffer was deterministically calculated, and M was deterministic. Lastly, it is clear that if M is an eDCM machine, then so is the resulting machine. □

Then, let $L \in$ eDCM. Let $M = (k, Q, \Sigma, \triangleleft, \delta, q_0, F) \in \mathsf{eDCM}(k), k \geq 1$. Assume without loss of generality, that every counter of M is increased in every computation, and they do so immediately (from q_0) in every counter. Also, assume that every counter empties before switching to a final state. Let $T = \{t_1, \ldots, t_m\}$ be labels in bijective correspondence with transitions of δ. Then, consider the following language L_M over alphabet $\Sigma \cup T$:

$$L_M = \{wx \mid \begin{array}{l}(q_0, w\triangleleft, 0, \ldots, 0) \vdash_M^{r_1} \cdots \vdash_M^{r_n} (q_n, \triangleleft, 0, \ldots, 0), q_n \in F, \\ w \in \Sigma^*, x \in T^*, x \text{ is the sequence of transitions on } \triangleleft\}.\end{array}$$

Then it is clear that $L_M \subseteq LT^*$, and that $L_M(T^*)^{-1} \cap \Sigma^* = L$.

Then given $M \in$ eDCM, L_M can be constructed using only inverse deterministic finite transducers, and commutative closure, as demonstrated next.

Proposition 22. *Given $M \in$ eDCM, the language L_M is in* eDCM *and L_M can be obtained from $\{\lambda\}$ by a combination of commutative closure and inverse deterministic finite transductions.*

Proof. It follows from Proposition 9 and Lemma 21 that eDCM is closed under taking commutative closure and inverse deterministic finite transductions. It will be shown that L_M can be obtained via a combination of commutative closure and inverse deterministic finite transductions. From this, it will follow that L_M is in eDCM. It can also be easily seen that every language family closed under inverse deterministic finite transductions is also closed under intersection with regular languages (the transducer simply outputs the input, and transitions according to a DFA accepting the regular language). Therefore, we will use intersection with regular languages as well.

From the language $\{\lambda\}$, with an inverse deterministic finite transduction, one can get any language Γ^* over an alphabet Γ.

Let M be defined as in the text preceding this proposition (with T, the transition labels). Then consider the language $R \subseteq T^*$ equal to

$$\{x \mid \ x \in T^*, (q_0, w\triangleleft, 0, \ldots, 0) \vdash_M^x (q_f, \triangleleft, 0, \ldots, 0), q_f \in F, w \in \Sigma^*\}.$$

Next, we will describe how to generate R from T^* using only commutative closure and intersection with regular languages. First, let $R_0 = T^*$. Then, for each i, $1 \leq i \leq k$, let T_{i+} (respectively, $T_{i-}, T_{i=}$) be the subsets of T that increase counter i by 1 (respectively decrease by 1, do not change counter i). Then, for each $i, 1 \leq i \leq k$, define R_i inductively as:

$$R_i = \text{comm}(R_{i-1}) \cap T_{i=}^*(T_{i-}T_{i+}T_{i=}^*)^+.$$

Then, let $R' = \text{comm}(R_k)$. It will be shown within the next claim that $R' = \{x \mid x \in T^+, \text{ where for all } i, 1 \leq i \leq k, |x|_{T_{i+}} = |x|_{T_{i-}} > 0\}$. In other words, the number of additions to each counter is the same as the number of deletions.

Let R'' be obtained from R' by intersecting with a regular language enforcing that:

- a single letter representing a transition from the initial state on all counters being 0 is read,
- followed by, for every i, $1 \leq i \leq k$, transitions on counter i being positive that do not decrease, followed by those on counter i being positive that do not increase, followed by those transitions on counter i being zero,
- the states transition as in M (ie. transitions switch states as in M, with a stay transition on a implying the next transition is on a as well),
- the last transition ends in a final state.

Note that this implies that in R'', the end-marker is reached before any decrease (since M is in eDCM), and once a transition label on the end-marker is reached, all further transition labels read are stay transitions on the end-marker.

Claim 1. $R = R''$.

Proof. Let $w \in R$. Then, $w = r_1 r_2 \cdots r_n$, where $(q_0, v\triangleleft, 0, \ldots, 0) \vdash_M^{r_1} \cdots \vdash_M^{r_n} (q_f, \triangleleft, 0, \ldots, 0), q_f \in F, v \in \Sigma^*$. Then, within $r_1, \ldots, r_n$, for each i, $1 \leq i \leq k$, there are as many transitions that increase counter i (a positive number) as those that decrease it. Hence, $w \in R'$. Then, the states corresponding to transitions applied in w change according to M, ending in a final state, the initial transition on q_0 and all counters being zero must occur immediately, followed by, for each i, $1 \leq i \leq k$, a sequence of transitions on counter i being positive that do not decrease, followed by a sequence that does not increase, followed by a sequence with counter i being 0. Hence, $w \in R''$.

Let $w = r_1 r_2 \cdots r_n$, where $r_j \in T, 1 \leq j \leq n$ be in R''. Then $R'' \subseteq R'$, and thus w has the same positive number of increasing transitions on counter i as decreasing transitions. Then, from the definition of R'', the states of w must change according to M, with an initial transition on all counters being zero, followed by, for each i, $1 \leq i \leq k$, transitions on i positive that are non-decreasing, followed by those on a positive counter that are not increasing, followed by transitions on counter i being 0, and finishing in a final state, with the end-marker appearing before any decrease. Thus, $(q_0, v\triangleleft, 0, \ldots, 0) \vdash_M^{r_1} \cdots \vdash_M^{r_n} (q_f, \triangleleft, 0, \ldots, 0), q_f \in F, v \in \Sigma^*$, and $w \in R$. □

Hence, it is possible to obtain R from T^* by only commutative closure and intersection with regular languages. Next, we will derive L_M from R via an inverse deterministic finite transduction. Let $A = (P, \Sigma \cup T, T, \triangleleft, \delta_A, p_0, F_A)$ be a deterministic finite transducer, such that A operates as follows on an input of $a_1 \cdots a_n r_1 \cdots r_m, n, m \geq 0, a_i \in \Sigma, r_j \in T, 1 \leq i \leq n, 1 \leq j \leq m$: On the first part of the input $a_1 \cdots a_n$, A simulates transitions of M with the first transition on 0 on every counter, followed by all positive counter transitions deterministically, while keeping track of the current state of M. So, if M switches from state q to p while moving right on an $a \in \Sigma$ (call this transition t), then A switches from state q to p while moving right on a, and outputting t. If t stays, then A does as well while outputting t. When reading $r_1 \cdots r_m$, A verifies that they are all transitions on $\triangleleft$, and A outputs $r_1 \cdots r_m$ deterministically.

Claim 2. $A^{-1}(R) = L_M$.

PROOF. Let $wx \in A^{-1}(R)$, where $w = a_1 \cdots a_n, n \geq 0, a_i \in \Sigma, 1 \leq i \leq n, x = r_1 \cdots r_m, m \geq 0, r_j \in T, 1 \leq j \leq m$. Then $(p_0, wx\lhd, \lambda) \vdash_A^* (q_f, \lhd, y), y \in R, q_f \in F_A$. Because $y \in R$, and as transition sequence y reads input w in M, by the construction of A, then $(q_0, w\lhd, 0, \ldots, 0) \vdash_M^y (q'_f, \lhd, 0, \ldots, 0), q'_f \in F$. Let $y = y_1 x$, which must be the case since A outputs every symbol from T verbatim. Also $r_1, \ldots, r_m$ must all be on $\lhd$ by the construction of A. And all of y_1 must not be on $\lhd$, since $a_1 \cdots a_n$ was read to output each symbol of y_1, and $a_1 \cdots a_n \in \Sigma^*$. Hence, $wx \in L_M$.

Let $wx \in L_M$, where $(q_0, w\lhd, 0, \ldots, 0) \vdash_M^{r_1} \cdots \vdash_M^{r_n} (q_n, \lhd, 0, \ldots, 0), q_n \in F, w \in \Sigma^*, x \in T^*$, x is the sequence of transitions on $\lhd$. Let y be such that $yx = r_1 \cdots r_n$. Then no transition of y is on $\lhd$, but each of x must be on $\lhd$. So $yx \in R$. Then, on input wx, A outputs y as it simulates M before $\lhd$, then on x, it outputs x. Hence, $wx \in A^{-1}(R)$. □

Hence, $L_M \in$ eDCM, and it can be obtained from $\{\lambda\}$ via a combination of commutative closure and inverse deterministic finite transducers. □

In [14], there is a construction that shows that DCM is closed under right quotient with NPCM. From $M_1 \in$ DCM, the construction creates a new DCM machine M' by simulating the original DCM machine M_1, while storing the Parikh vector of the input in a set of additional counters, and then performing some additional processing after reading the input. Thus, if the original machine is an eDCM machine, the resulting one is as well. Hence eDCM is also closed under right quotient with NPCM languages.

From L_M, taking right quotient with the regular language T^* and intersecting with Σ^* (every family closed under inverse deterministic finite transductions is also closed under intersection with regular languages) gives L. Hence, we obtain the following:

Proposition 23. eDCM *is the smallest family of languages (containing* $\{\lambda\}$*) that is closed under taking commutative closure, inverse deterministic finite transductions, and right quotient with regular languages.*

Thus, the characterization of eDCM in terms of commutative closure, right quotient, and inverse deterministic finite transducers is not enough to generate all DCM languages. There is a known, relatively simple characterization of DCM in terms of closure properties whereby DCM is the smallest family of languages closed under inverse deterministic finite transductions augmented by reversal-bounded counters [19]. However, it is still an open problem as to whether it is possible to characterize DCM using a more complicated set of closure properties.

Acknowledgements

We thank an anonymous reviewer of an earlier version of this paper for suggesting the comparison between eDCM and deterministic Parikh automata.

[1] R. Parikh, On context-free languages, J. ACM 13 (4) (1966) 570–581.
[2] M. Harrison, Introduction to Formal Language Theory, Addison-Wesley, Reading, Ma, 1978.
[3] B. S. Baker, R. V. Book, Reversal-bounded multipushdown machines, Journal of Computer and System Sciences 8 (3) (1974) 315–332.
[4] O. H. Ibarra, Reversal-bounded multicounter machines and their decision problems, J. ACM 25 (1) (1978) 116–133.
[5] M. Latteux, Cônes rationnels commutatifs, Journal of Computer and System Sciences 18 (3) (1979) 307–333.
[6] M. Cadilhac, A. Finkel, P. McKenzie, Affine Parikh automata, RAIRO - Theoretical Informatics and Applications 46 (2012) 511–545.
[7] S. Crespi-Reghizzi, P. S. Pietro, Commutative languages and their composition by consensual methods, in: Proceedings 14th International Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014., 2014, pp. 216–230.
[8] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, MA, 1979.
[9] C. Papadimitriou, Computational complexity, Addison-Wesley, Reading, Massachusetts, 1994.
[10] O. Ibarra, Automata with reversal-bounded counters: A survey, in: H. Jürgensen, J. Karhumäki, A. Okhotin (Eds.), Descriptional Complexity of Formal Systems, Vol. 8614 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 5–22.
[11] M. Cadilhac, A. Finkel, P. McKenzie, Bounded Parikh automata, International Journal of Foundations of Computer Science 23 (08) (2012) 1691–1709.

[12] J. Eremondi, O. Ibarra, I. McQuillan, Insertion operations on deterministic reversal-bounded counter machines, in: A. Dediu, E. Formenti, C. Martín-Vide, B. Truthe (Eds.), Lecture Notes in Computer Science, Vol. 8977 of 9th International Conference on Language and Automata Theory and Applications, LATA 2015, Nice, France, 2015, pp. 200–211.
[13] S. Ginsburg, S. Greibach, Deterministic context free languages, Information and Control 9 (6) (1966) 620–648.
[14] J. Eremondi, O. Ibarra, I. McQuillan, Deletion operations on deterministic families of automata, in: R. Jain, S. Jain, F. Stephan (Eds.), Lecture Notes in Computer Science, Vol. 9076 of 12th Annual Conference on Theory and Applications of Models of Computation, TAMC 2015, Singapore, 2015, pp. 388–399.
[15] O. Ibarra, S. Seki, Characterizations of bounded semilinear languages by one-way and two-way deterministic machines, International Journal of Foundations of Computer Science 23 (6) (2012) 1291–1306.
[16] C. Nicaud, Average state complexity of operations on unary automata, in: M. Kutyłowski, L. Pacholski, T. Wierzbicki (Eds.), Mathematical Foundations of Computer Science 1999, Vol. 1672 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1999, pp. 231–240.
[17] T. Harju, O. Ibarra, J. Karhumäki, A. Salomaa, Some decision problems concerning semilinearity and commutation, Journal of Computer and System Sciences 65 (2002) 278–294.
[18] R. E. Stearns, J. Hartmanis, P. M. Lewis, Hierarchies of memory limited computations, in: Proceedings of the 6th Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1965), FOCS '65, IEEE Computer Society, Washington, DC, USA, 1965, pp. 179–190.
[19] J. Eremondi, O. Ibarra, I. McQuillan, Insertion operations on deterministic reversal-bounded counter machines, journal version of [12] submitted (2015).