The University of Saskatchewan
Department of Computer Science

Technical Report #2016-01

UNIVERSITY OF
SASKATCHEWAN

On Families of Full Trios Containing Counter Machine Languages *

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

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

Abstract

We look at NFAs augmented with multiple reversal-bounded counters where, during an accepting computation, the behavior of the counters during increasing and decreasing phases is specified by some fixed "pattern". We consider families of languages defined by various pattern behaviors and show that some correspond to the smallest full trios containing restricted classes of bounded semilinear languages. For example, one such family is exactly the smallest full trio containing all the bounded semilinear languages. Another family is the smallest full trio containing all the bounded context-free languages. Still another is the smallest full trio containing all bounded languages whose Parikh map is a semilinear set where all periodic vectors have at most two non-zero coordinates. We also examine relationships between the families.

1 Introduction

A language L is bounded if $L \subseteq w_1^* \cdots w_k^*$, for non-empty words $w_1, \ldots, w_k$. Further, L is *bounded semilinear* if there exists a semilinear set $Q \subseteq \mathbb{N}_0^k$ such that $L = \{w \mid w = w_1^{i_1} \cdots w_k^{i_k}, (i_1, \ldots, i_k) \in Q\}$ [10]. It is known that every bounded semilinear language can be accepted by a one-way nondeterministic reversal-bounded multicounter machine (NCM, [9]). Also, every bounded language accepted by an NCM can be accepted by a deterministic NCM (DCM, [10]). Thus, every bounded semilinear language can be accepted by a DCM.

Recently, several families of languages that are both bounded and semilinear have been defined and studied [7]. The notion of bounded semilinear above is referred to as *bounded Ginsburg semilinear* to distinguish from other types. Two other interesting types are: a language $L \subseteq w_1^* \cdots w_k^*$ is *bounded Parikh semilinear* if $L = \{w \mid w = w_1^{i_1} \cdots w_k^{i_k}$, the Parikh map of w is in $Q\}$, where Q is a semilinear set with $|\Sigma|$ components; L is *bounded general semilinear* if L is both bounded and semilinear. It was shown that the family of bounded Parikh semilinear languages is a strict subset of the family of bounded Ginsburg semilinear languages, which is a strict subset of the family of bounded general semilinear languages. However, it was shown that in any language family $\mathcal{L}$ that is a semilinear trio (the family only contains semilinear languages, and is closed under λ-free homomorphism, inverse homomorphism, and intersection with regular languages), all bounded languages within $\mathcal{L}$ are bounded Ginsburg semilinear, and are therefore in NCM and even DCM, enabling

*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 Natural Sciences and Engineering Research Council of Canada Grant 327486-2010.

many decidability properties for bounded languages in $\mathcal{L}$. Furthermore, a criteria was developed for testing when the bounded languages within $\mathcal{L}$ and DCM coincide; this occurs if and only if $\mathcal{L}$ contains all distinct-letter-bounded Ginsburg semilinear languages. This was shown to be the case for finite-index $\mathsf{ET0L}$ languages [12], and therefore the bounded languages within these families are the same.

In this paper, we attempt to restrict the operation of NCM in order to precisely characterize types of languages that are bounded and semilinear. Indeed, restricting the behavior of NCM can naturally capture several interesting families of bounded languages. This is accomplished through the use of so-called *instruction languages*. Informally, a k-counter machine M is said to satisfy instruction language $I \subseteq \{C_1, D_1, \dots, C_k, D_k\}^*$ if, for every accepting computation of M, replacing each increase of counter i with C_i, and decrease of counter i with D_i, gives a sequence in I. Then, for a family of instruction languages $\mathcal{I}$, $\mathsf{NCM}(\mathcal{I})$ is the family of NCM machines satisfying some $I \in \mathcal{I}$. Several interesting instruction language families are defined and studied. For example, if one considers $\mathsf{BD}_i\mathsf{LB}_d$, the family of instruction languages consisting of bounded increasing instructions followed by letter-bounded decreasing instructions, then we show that the family of languages accepted by $\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d)$ is the smallest full trio containing all bounded Ginsburg semilinear languages (and therefore, the smallest full trio containing all bounded languages from any semilinear trio). It is also possible to characterize exactly the bounded context-free languages with a subfamily of counter languages. Several other families are also defined and compared. For each, characterizations are given with a single language for each number of counters such that the families are the smallest full trios containing the languages. Using these characterizations, we are able to give even simpler criteria than those in [7] for testing if the bounded languages within a semilinear full trio coincide with those in DCM. We then give applications to several interesting families, such as the multi-pushdown languages [1], and restricted types of Turing machines, and it is shown that the bounded languages within each are the same as those accepted by DCM. In a future paper, we will examine closure and decision properties of the models.

2 Preliminaries

In this paper, we assume knowledge of automata and formal languages, and refer to [6] for an introduction. Let Σ be a finite alphabet. Then, Σ^* (resp. Σ^+) is the set of all words (non-empty words) over Σ. A word is any $w \in \Sigma^*$, and a language is any $L \subseteq \Sigma^*$. The empty word is denoted by λ. The complement of L with respect to Σ, $\overline{L} = \Sigma^* - L$. The shuffle of words $u, v \in \Sigma^*$, $u \sqcup\!\sqcup\, v = \{u_1 v_1 \cdots u_n v_n \mid u = u_1 \cdots u_n, v = v_1 \cdots v_n, u_i, v_i \in \Sigma^*, 1 \le i \le n\}$, extended to languages $L_1 \sqcup\!\sqcup\, L_2 = \{u \sqcup\!\sqcup\, v \mid u \in L_1, v \in L_2\}$.

A language $L \subseteq \Sigma^*$ is bounded if there exists $w_1, \dots, w_k \in \Sigma^+$ such that $L \subseteq w_1^* \cdots w_k^*$, and is letter-bounded if $w_1, \dots, w_k$ are letters. Furthermore, L is distinct-letter-bounded if each letter is distinct.

Let $\mathbb{N}$ be the set of positive integers and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. A *linear set* is a set $Q \subseteq \mathbb{N}_0^m$ if there exists $\vec{v_0}, \vec{v_1}, \dots, \vec{v_n}$ such that $Q = \{\vec{v_0} + i_1\vec{v_1} + \cdots + i_n\vec{v_n} \mid i_1, \dots, i_n \in \mathbb{N}_0\}$. The vector $\vec{v_0}$ is called the *constant*, and $\vec{v_1}, \dots, \vec{v_n}$ are the *periods*. A *semilinear set* is a finite union of linear sets. Given an alphabet $\Sigma = \{a_1, \dots, a_m\}$, the length of a word $w \in \Sigma^*$ is denoted by $|w|$. And, given $a \in \Sigma$, $|w|_a$ is the number of a's in w. Then, the *Parikh map* of w is $\psi(w) = (|w|_{a_1}, \dots, |w|_{a_m})$, and the Parikh map of a language L, $\psi(L) = \{\psi(w) \mid w \in L\}$. Also, $\mathrm{alph}(w) = \{a \in \Sigma \mid |w|_a > 0\}$. We refer to Section 1 for the definitions of bounded Ginsburg semilinear and bounded Parikh semilinear languages.

For a class of machines $\mathcal{M}$, we let $\mathcal{L}(\mathcal{M})$ be the family of languages accepted by machines in

$\mathcal{M}$. Let $\mathcal{L}(\mathsf{CFL})$ be the family of context-free languages. A *trio* (resp. *full trio*) is any family of languages closed under λ-free homomorphism (resp. homomorphism), inverse homomorphism, and intersection with regular languages. A *full semi-AFL* is a full trio closed under union [2]. Given a language family $\mathcal{L}$, $\mathcal{L}^{\text{bd}}$ are the bounded languages in $\mathcal{L}$.

A *one-way* k-counter machine [9] is a tuple $M = (k, Q, \Sigma, \lhd, \delta, q_0, F)$, where $Q, \Sigma, \lhd, q_0, F$ are respectively the finite set of states, input alphabet, right input end-marker (unnecessary for nondeterministic machines and will largely not be used in this paper), initial state, and final states. The transition relation is a relation from $Q \times (\Sigma \cup \{\lhd, \lambda\}) \times \{0,1\}^k$ to $Q \times \{-1, 0, +1\}^k$, such that $(p, d_1, \ldots, d_k) \in \delta(q, a, c_1, \ldots, c_k)$ and $c_i = 0$ implies $d_i \geq 0$ to prevent negative values in the counters. Also, M is deterministic if $|\delta(q, a, c_1, \ldots, c_k) \cup \delta(q, \lambda, c_1, \ldots, c_k)| \leq 1$, for all $q \in Q, a \in \Sigma \cup \{\lhd\}, (c_1, \ldots, c_k) \in \{0,1\}^k$. A configuration of M is a tuple $(q, w, i_1, \ldots i_k)$ where q is the current state, $w \in \Sigma^* \lhd \cup \{\lambda\}$ is the remaining input, and $i_1, \ldots, i_k$ are the contents of the counters. The derivation relation $\vdash_M$ is defined between configurations, where $(q, aw, i_1, \ldots, i_k) \vdash_M (p, w, i_1 + d_1, \ldots, i_k + d_k)$ if there is a transition $(p, d_1, \ldots, d_k) \in \delta(q, a, c_1, \ldots, c_k)$, where c_j is 1 if $i_j > 0$, and $c_j = 0$ otherwise, if $i_j = 0$. Let $\vdash_M^*$ the reflexive, transitive closure of $\vdash_M$. M accepts a word $w \in \Sigma$ if $(q_0, w\lhd, 0, \ldots, 0) \vdash_M^* (q_f, \lambda, i_1, \ldots, i_k), q_f \in F, i_1, \ldots, i_k \in \mathbb{N}_0$, and the language of all words accepted by M is denoted by $L(M)$.

Further, M is l-reversal-bounded if, in every accepting computation, the counter alternates between increasing and decreasing at most l times. We will often associate labels from an alphabet T to the transitions of M bijectively, and then write $\vdash_M^t$ to represent the changing of configurations via transition t. This is generalized to derivations over words in T^*.

Then $\mathsf{NCM}(k, l)$ is the class of one-way l-reversal-bounded k-counter machines, and NCM is all reversal-bounded multicounter languages, and replacing N with D gives the deterministic variant.

3 Instruction NCM Machines

It is known that the bounded languages in $\mathcal{L}(\mathsf{NCM})$ are a "limit" to the bounded languages in semilinear trios [7]. We start this section by considering subclasses of $\mathcal{L}(\mathsf{NCM})$ in order to determine more restricted methods of computation that can also form such a limit. We are able to do this optimally. Furthermore, characterizations of the restricted families are also possible, and lead to even simpler methods to determine the bounded languages within semilinear full trios.

First, we define restrictions of NCM depending on the sequences of counter instructions that occur. These restrictions will only be defined on NCMs that we will call *well-formed*. A k-counter NCM M is well-formed if $M \in \mathsf{NCM}(k, 1)$ whereby all transitions change at most one counter value per transition, and all counters decrease to zero before accepting. Indeed, an NCM (or DCM) can be assumed without loss of generality to be 1-reversal-bounded by increasing the number of counters [9]. It is also clear that all counters can be forced to change one counter value at a time, and decrease to zero without loss of generality. Thus, every language in $\mathcal{L}(\mathsf{NCM})$ can be accepted by a well-formed NCM. Also, since we will only be considering nondeterministic machines, we will not include $\lhd$. Let Δ be an infinite set of new symbols, $\Delta = \{C_1, D_1, C_2, D_2, \ldots\}$, and for $k \geq 1, \Delta_k = \{C_1, D_1, \ldots, C_k, D_k\}, \Delta_{(k,c)} = \{C_1, \ldots, C_k\}, \Delta_{(k,d)} = \{D_1, \ldots, D_k\}$.

Given a well-formed k-counter NCM machine M, let T be a set of labels in bijective correspondence with transitions of M. Then, define a homomorphism h_Δ from T^* to Δ_k that maps every transition label associated with a transition that increases counter i to C_i, maps every label associated with a transition that decreases counter i to D_i, and maps all labels associated with transitions that do not change any counter to λ. Also, define a homomorphism h_Σ that maps every transition that reads a letter $a \in \Sigma$ to a, and erases all others. Then, we say that M *satisfies instruction* *language* $I \subseteq \Delta_k^*$ if every sequence of transitions $\alpha \in T^*$ corresponding to an accepting computation — that is $(q_0, w, 0, \dots, 0) \vdash_M^{\alpha} (q, \lambda, c_1, \dots, c_k), q$ a final state — has $h_\Delta(\alpha) \in I$. This means that M satisfies instruction language I if I describes all possible counter increase and decrease instructions that can be performed in an accepting computation by M, with C_i occurring for every increase of counter i by one, and D_i occurring for every decrease of counter i by one.

Given a family of languages $\mathcal{I}$ with each $I \in \mathcal{I}$ over Δ_k, for some $k \geq 1$, let $\mathsf{NCM}(k, \mathcal{I})$ be the subset of well-formed k-counter NCM machines that satisfy I for some $I \in \mathcal{I}$ with $I \subseteq \Delta_k^*$; these are called the k-counter $\mathcal{I}$-instruction machines. The family of languages they accept, $\mathcal{L}(\mathsf{NCM}(k, \mathcal{I}))$, are called the k-counter $\mathcal{I}$-instruction languages. Furthermore, $\mathsf{NCM}(\mathcal{I}) = \bigcup_{k \geq 1} \mathsf{NCM}(k, \mathcal{I})$ (resp. $\mathcal{L}(\mathsf{NCM}(\mathcal{I}) = \bigcup_{k \geq 1} \mathcal{L}(\mathsf{NCM}(k, \mathcal{I}))$ are the $\mathcal{I}$-instruction machines (and languages). We will only consider instruction languages I where, for all $w \in I$, every occurrence of C_i occurs before any occurrence of D_i, for all i, $1 \leq i \leq k$, which is enough since every well-formed machine is 1-reversal-bounded.

First, we will study properties of these restrictions before examining some specific types.

Proposition 1. *Given any family of languages $\mathcal{I}$ over Δ_k, $\mathcal{L}(\mathsf{NCM}(k, \mathcal{I}))$ is a full trio. Furthermore, given any family of languages $\mathcal{I}$, where each $I \in \mathcal{I}$ is over some $\Delta_k, k \geq 1$, $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ is a full trio.*

Proof. The standard proofs for closure under homomorphism and inverse homomorphism apply. The proof for intersection with regular languages also works, as restricting the words of the language can restrict the possible sequences of instructions appearing in accepting computations, but the resulting sequences of instructions will therefore be a subset of the instruction language of the original machine. □

Next, we require another definition. Given a language I over Δ_k, let

$$I_{eq} = \{w \mid w \in I, |w|_{C_i} = |w|_{D_i}, \text{every } C_i \text{ occurs before any } D_i, \text{ for } 1 \leq i \leq k\}.$$

Further, given a language family $\mathcal{I}$ over Δ where each $I \in \mathcal{I}$ is over Δ_k, for some $k \geq 1$, then $\mathcal{I}_{eq}$ is the family of all languages I_{eq}, where $I \in \mathcal{I}$.

Proposition 2. *Let $\mathcal{I}$ be a family of languages where each $I \in \mathcal{I}$ is a subset of Δ_k^*, for some $k \geq 1$, and $\mathcal{I}$ is a subfamily of the regular languages. Then $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ is the smallest full trio containing $\mathcal{I}_{eq}$.*

Proof. First, it follows from Proposition 1 that $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ is a full trio.

To see that $\mathcal{I}_{eq} \subseteq \mathcal{L}(\mathsf{NCM}(\mathcal{I}))$, let $I \in \mathcal{I}$ and let M be a DFA accepting $I \subseteq \Delta_k^*$. Then we will create a well-formed k-counter machine M' that accepts I_{eq} as follows: M' simulates M while adding to counter i for every C_i read, and subtracting from counter i for every D_i read (never adding after subtracting), accepting if M does, and if all counters end at zero. Then M' accepts all words of I with an equal number of C_i's as D_i's, for each i where all C_i's occur before any D_i. This is exactly I_{eq}. Also, M' satisfies I_{eq} and I. Hence $\mathcal{I}_{eq} \subseteq \mathcal{L}(\mathsf{NCM}(\mathcal{I}))$.

Next we will verify that $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ is the smallest full trio containing $\mathcal{I}_{eq}$. For this, let $M = (k, Q, \Sigma, \lhd, \delta, q_0, F) \in \mathsf{NCM}(\mathcal{I})$ with k counters that satisfies instruction language $I \in \mathcal{I}$.

Let g be a homomorphism from Γ^* to Δ_k^* (Γ defined below) that maps (q, a, X_i, p) to X_i, where $X_i \in \Delta_k, p, q \in Q, a \in \Sigma \cup \{\lambda\}$, and there is a transition from q to p on a that increases counter i if $X_i = C_i$, decreases counter i if $X_i = D_i$; similarly g maps $(q, a, 0, p)$ to λ, where $p, q \in Q, a \in \Sigma \cup \{\lambda\}$, where there is a transition from q to p on a that does not change any counter. Both of these types of symbols can be created from transitions defined on any counter value (0 or positive). We say that symbol $(q, a, X_i, p), X_i \in \Delta_k \cup \{0\}$ is defined on counter i positive if it was created above from a transition defined on counter i being positive. We say that the symbol is defined on counter i being zero if it was created from a transition on counter i being zero (such a symbol could be defined on counter i being both 0 and positive).

Then, create a regular language $R \subseteq \Gamma^*$, $R = \{y_0 y_1 \cdots y_n \mid y_i = (p_i, a_{i+1}, X_{i+1}, p_{i+1}), p_0 = q_0, p_{n+1} \in F$, for each i, $1 \le i \le k$, if j is the smallest such that $X_j = C_i$ and if l is the largest such that $X_j = D_i$, then $y_0, \ldots, y_j$ are defined on counter i zero, $y_{j+1}, \ldots, y_l$ are defined on counter i positive, and $y_{l+1}, \ldots, y_n$ are defined on counter i zero$\}$.

Let h be a homomorphism from Γ^* to Σ^* such that h projects onto the second component. Then it is clear that $L(M) = h(g^{-1}(I_{eq}) \cap R)$ since $g^{-1}(I_{eq}) \cap R$ consists of all words of R with an equal number of C_i's as D_i's, for each i, $1 \le i \le k$. □

We will consider several instruction language families that define interesting subfamilies of $\mathcal{L}(\mathsf{NCM})$.

Definition 1. *We define instruction language families:*

- $\mathsf{LB}_i\mathsf{LB}_d = \{I = YZ \mid k \ge 1, Y = a_1^* \cdots a_m^*, a_i \in \Delta_{(k,c)}, 1 \le i \le m, Z = b_1^* \cdots b_n^*, b_j \in \Delta_{(k,d)}, 1 \le j \le n\}$, *(letter-bounded-increasing/letter-bounded-decreasing instructions),*
- $\mathsf{StLB}_{id} = \{I \mid k \ge 1, I = a_1^* \cdots a_m^*, a_i \in \Delta_k, 1 \le i \le m,$ *there is no* $1 \le l < l' < j < j' \le m$ *such that* $a_l = C_r, a_{l'} = C_s, a_j = D_r, a_{j'} = D_s, r \ne s\}$, *(stratified-letter-bounded instructions),*
- $\mathsf{LB}_{id} = \{I \mid k \ge 1, I = a_1^* \cdots a_m^*, a_i \in \Delta_k, 1 \le i \le m\}$, *(letter-bounded instructions),*
- $\mathsf{BD}_i\mathsf{LB}_d = \{I = YZ \mid k \ge 1, Y = w_1^* \cdots w_m^*, w_i \in \Delta_{(k,c)}^*, 1 \le i \le m, Z = a_1^* \cdots a_n^*, a_j \in \Delta_{(k,d)}, 1 \le j \le n\}$, *(bounded-increasing/letter-bounded-decreasing instructions),*
- $\mathsf{LB}_i\mathsf{BD}_d = \{I = YZ \mid k \ge 1, Y = a_1^* \cdots a_m^*, a_i \in \Delta_{(k,c)}, 1 \le i \le m, Z = w_1^* \cdots w_n^*, w_j \in \Delta_{(k,d)}^*, 1 \le j \le n\}$, *(letter-bounded-increasing/bounded-decreasing instructions),*
- $\mathsf{BD}_{id} = \{I \mid k \ge 1, I = w_1^* \cdots w_m^*, w_i \in \Delta_k^*, 1 \le i \le m\}$, *(bounded instructions),*
- $\mathsf{LB}_d = \{I \mid k \ge 1, I = Y \sqcup\!\sqcup Z, Y = \Delta_{(k,c)}^*, Z = a_1^* \cdots a_n^*, a_j \in \Delta_{(k,d)}, 1 \le j \le n\}$, *(letter-bounded-decreasing instructions),*
- $\mathsf{LB}_i = \{I \mid k \ge 1, I = Y \sqcup\!\sqcup Z, Y = a_1^* \cdots a_m^*, a_i \in \Delta_{(k,c)}, 1 \le i \le m, Z = \Delta_{(k,d)}^*, \}$, *(letter-bounded increasing instructions),*
- $\mathsf{LB}_\cup = \mathsf{LB}_d \cup \mathsf{LB}_i$, *(either letter-bounded-decreasing or letter-bounded-increasing instructions),*
- $\mathsf{ALL} = \{I \mid k \ge 1, I = \Delta_k^*\}$.

For example, every NCM machine M where the counters are increased and decreased according to some bounded language, then there is an instruction language I such that M satisfies I, and $I \in \mathsf{BD}_{id}$, and $L(M) \in \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$. Even though not all instructions in I are necessarily used, the instructions used will be a subset of I since the instructions used are a subset of a bounded language. It is also clear that $\mathcal{L}(\mathsf{NCM}) = \mathcal{L}(\mathsf{NCM}(\mathsf{ALL}))$.

Example 1. *Let* $L = \{ua^i vb^j wa^i xb^j y \mid i, j > 0, u, v, w, x, y \in \{0,1\}^*\}$. *We can easily construct a well-formed 2-counter machine* M *to accept* L *where, on input* $ua^i vb^j wa^{i'} xb^{j'} y$, M *increases counter 1* i *times, then increases counter 2* j *times, then decreases counter 1 verifying that* $i = i'$, *then decreases counter 2 verifying that* $j = j'$. *This machine satisfies instruction language* $C_1^* C_2^* D_1^* D_2^*$, *which is a subset of some instruction language in every family in Definition 1 except for* StLB_{id}, *and therefore* $L \in \mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ *for each of these families* $\mathcal{I}$.

Example 2. *Let $L = \{a^{2+i+2j}b^{3+2i+5j} \mid i, j \geq 0\}$. Note that the Parikh map of L is a linear set $Q = \{(2,3)+(1,2)i+(2,5)j \mid i, j \geq 0\}$. L can be accepted by a well-formed 4-counter* NCM *M as follows, when given input $a^m b^n$: first, on λ-moves, M increments counters 1 and 2 a nondeterministically guessed number of times $i \geq 0$, then on λ-moves, increments counters 3 and 4 a nondeterministically guessed number of times $j \geq 0$. Then, M verifies that $m = 2 + i + 2j$ by reading 2 a's and using (i.e., decrementing) counter 1 to zero and then 3 to zero to check that the remaining number of a's is equal to the value of counter 1 plus 2 times the value of counter 3. Finally, M checks and accepts if $n = 3 + 2i + 5j$ by first reading 3 b's and decrementing counter 2 and then 4. The instructions of M as constructed are a subset of $I = (C_1C_2)^*(C_3C_4)^*D_1^*D_3^*D_2^*D_4^*$. This is a subset of some language in each of $\mathsf{BD}_i\mathsf{LB}_d, \mathsf{BD}_{id}, \mathsf{LB}_d$ but not the other families, and therefore M is in each of $\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d), \mathsf{NCM}(\mathsf{BD}_{id}), \mathsf{NCM}(\mathsf{LB}_d)$. Even though M is not in the other classes of machines such as $\mathsf{NCM}(\mathsf{LB}_i)$, it is possible for $L(M)$ to be in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i))$ (using some other machine that accepts the same language). Indeed, we will see that $L(M)$ is also in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d))$ and $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i))$.*

Example 3. *Let $L_1 = \{w\#a^ib^j \mid |w|_a = i, |w|_b = j\}$. We construct M_1 that reads w, and adds to the first counter for every a read, and adds to the second counter for every b read. Then, after the $\#$ symbol, M_1 subtracts from counter 1 for every a read, then when reaching a b, it switches to decreasing the second counter for every b read. Therefore, it satisfies language $\{C_1, C_2\}^*D_1^*D_2^*$ which is indeed a subset of $\{C_1, C_2\}^* \sqcup\!\sqcup D_1^*D_2^* \in \mathsf{LB}_d$.*

Now let $L_2 = L_1^R$. We conjecture that L_2 is not in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_d))$, but we can construct a machine $M_2 \in \mathsf{NCM}(\mathsf{LB}_i)$ to accept L_2. (M_2, when given $b^ja^i\#w$, stores i and j in two counters and then checks by decrementing the counters that $|w|_a = i$ and $|w|_b = j$.) Similarly, we conjecture that L_1 is not in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i))$. Obviously, L_1 and L_2 are both in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_\cup))$.

Example 4. *Let $L = \{w \mid w \in \{a,b\}^+, |w|_a = |w|_b > 0\}$. L can be accepted by an* NCM *which uses two counters that increments counter 1 (resp. counter 2) whenever it sees an a (resp., b). Then it decrements counter 1 and counter 2 simultaneously and accepts if they reach zero at the same time. This counter usage does not have a pattern in any of the restrictions above. It is quite unlikely that $L(M) \in \mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ for any of the families in the definition above except the full $\mathcal{L}(\mathsf{NCM}(\mathsf{ALL})) = \mathcal{L}(\mathsf{NCM})$.*

Every family $\mathcal{I}$ in Definition 1 is a subfamily of the regular languages. Therefore, by Proposition 2, the following can be shown by proving closure under union:

Proposition 3. *Let $\mathcal{I}$ be any family of instruction languages from Definition 1. Then $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ is the smallest full trio (and full semi-AFL) containing $\mathcal{I}_{eq}$.*

Proof. It suffices to show closure under union for each family. For $\mathsf{LB}_i\mathsf{LB}_d$, given two machines M_1, M_2 with k_1, k_2 counters respectively, we can build a $k_1 + k_2$ counter machine M where M adds to counters according to M_1 using the first k_1 counters, then decreasing according to M_1, or the same with M_2 on the remaining counters. It is clear that this gives an instruction language that is a subset of the increasing pattern of M_1 followed by the increasing pattern of M_2, followed by the decreasing pattern of M_2, then M_1. This is letter-bounded insertion followed by letter-bounded deletion behavior. The same construction works in all other cases. $\square$

As a corollary, if we consider the instructions languages of ALL (thus, the instructions are totally arbitrary), and for $i \geq 1$, let $L_i = \{C_i^nD_i^n \mid n \geq 0\}$, then $\mathsf{ALL}_{eq} = \{I \mid I = L_1 \sqcup\!\sqcup L_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup L_k, k \geq 1\}$. Hence, $\mathcal{L}(\mathsf{NCM})$ can be characterized as the smallest full trio containing ALL_{eq}, by Proposition 2. Or, it could be stated as follows (this is essentially already known, and follows from work in [4] and [5]).

Corollary 1. *[4, 5]* $\mathcal{L}(\mathsf{NCM})$ *is the smallest shuffle or intersection closed full trio containing* $\{a^n b^n \mid n \geq 0\}$.

Indeed, it is known that $\mathcal{L}(\mathsf{NCM})$ is shuffle and intersection closed full trio [9]. For intersection, this follows since each instruction language I above can be represented by taking each L_i, and a homomorphism h_i that maps C_i and D_i to itself, and erases all other letters of Δ_k. Then let $L_i' = h_i^{-1}(L_i)$. Then, $L_1 \sqcup\!\sqcup L_2 \sqcup\!\sqcup \cdots \sqcup\!\sqcup L_k = L_1' \cap L_2' \cap \cdots \cap L_k'$.

Since $\{a^n b^n \mid n \geq 0\}$ is in $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ for all $\mathcal{I}$ in Definition 1, the following is also immediate from Corollary 1:

Corollary 2. *For all* $\mathcal{I}$ *in Definition 1,* $\mathcal{L}(\mathsf{NCM})$ *is the smallest shuffle or intersection closed full trio containing* $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$.

Thus, any instruction family $\mathcal{I}$ whereby $\mathcal{L}(\mathsf{NCM}(\mathcal{I})) \subsetneq \mathcal{L}(\mathsf{NCM})$ and $\{a^n b^n \mid n \geq 0\} \in \mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ is immediately not closed under intersection and shuffle.

Next, we will prove the following lemma regarding many of the instruction language families showing that letter-bounded instructions can be assumed to be distinct-letter-bounded, and for bounded languages, for each letter in Δ_k in the words to only appear once.

First, we need a definition. For each of the instruction families of Definition 1, we place an underline below each LB if the letter-bounded language is forced to have each letter occur exactly once (and therefore be distinct-letter-bounded), and we place an underline below BD if each letter $a \in \Delta_k$ appears exactly once within the words $w_1, \ldots, w_m$. Thus, as an example, $\underline{\mathsf{LB}}_i\underline{\mathsf{BD}}_d$ is the subset of $\mathsf{LB}_i\mathsf{BD}_d$ equal to $\{I = YZ \mid k \geq 1, Y = a_1^* \cdots a_k^*, a_i \in \Delta_{(k,c)}, |a_1 \cdots a_k|_a = 1,$ for all $a \in \Delta_{(k,c)}, Z = w_1^* \cdots w_n^*, w_i \in \Delta_{(k,d)}^*, 1 \leq j \leq n, |w_1 w_2 \cdots w_n|_a = 1,$ for all $a \in \Delta_{(k,d)}\}$. Thus, each letter appears exactly once in the words or letters. The construction uses multiple new instruction letters and counters, in order to allow each letter to only appear once.

Lemma 1. *The following are true:*

$\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d)) = \mathcal{L}(\mathsf{NCM}(\underline{\mathsf{LB}}_i\underline{\mathsf{LB}}_d))$, $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id})) = \mathcal{L}(\mathsf{NCM}(\underline{\mathsf{LB}}_{id}))$,
$\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d)) = \mathcal{L}(\mathsf{NCM}(\underline{\mathsf{LB}}_i\underline{\mathsf{BD}}_d))$, $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_d)) = \mathcal{L}(\mathsf{NCM}(\underline{\mathsf{LB}}_d))$,
$\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d)) = \mathcal{L}(\mathsf{NCM}(\underline{\mathsf{BD}}_i\underline{\mathsf{LB}}_d))$, $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i)) = \mathcal{L}(\mathsf{NCM}(\underline{\mathsf{LB}}_i))$.

Proof. First, consider the case for LB_{id}. Let $M \in \mathsf{NCM}(k, \mathsf{LB}_{id})$ that satisfies $I \subseteq a_1^* \cdots a_n^*, a_i \in \Delta_k, 1 \leq i \leq n$. We will construct a machine $M' \in \mathsf{NCM}(m, \mathsf{LB}_{id})$ (with m potentially bigger than k) such that M' satisfies $I' \subseteq b_1^* \cdots b_{2m}^*$, where $b_1, \ldots, b_{2m}$ is some permutation of the symbols in Δ_m, and $L(M) = L(M')$.

We will proceed in steps, removing each counter x whereby at least one of C_x or D_x occurs multiple times in $a_1, \ldots, a_n$, one counter at a time. Then, for each such x, we convert M with I to M_x and $I_x \subseteq d_1^* \cdots d_l^*, d_1, \ldots, d_l \in \Delta_m$, whereby counter x will be removed, and multiple counters added back, and then each new symbol C_y and D_y only appears once within $d_1, \ldots, d_l$, such that M_x satisfies I_x, and $L(M) = L(M_x)$.

Let
$$f(x) = i_1, \ldots, i_\alpha, \tag{1}$$
be the sequence of all positions where $a_{i_p} = C_x$, $1 \leq p \leq \alpha$, and let
$$g(x) = j_1, \ldots, j_\beta, \tag{2}$$
be all positions where $a_{j_q} = D_x$, $1 \leq q \leq \beta$.

Then in M_x, instead of using counter x, M_x replaces it with $\alpha \cdot \beta$ counters, which we refer to by $(p,q), 1 \leq p \leq \alpha, 1 \leq q \leq \beta$, and we will use $C_{(p,q)}$ (resp. $D_{(p,q)}$) to represent the instruction character while increasing (resp. decreasing) counter (p,q) (these can easily be replaced with consecutive numbered characters in Δ_m at the end of the procedure). Then, M_x simulates M identically for all counters other than x. When simulating M increasing counter x in section i_p, for $1 \leq p \leq \alpha$, M_x instead uses and increases counter $(p,1)$ until some nondeterministically chosen spot where M_x switches to and increases from counter $(p,2)$ (while still simulating the same section i_p of M), etc., through counter (p,β). Then, when simulating the decrease of counter x in section j_q of M, for $1 \leq q \leq \beta$, instead, M_x decreases from counters $(1,q)$ until empty, then $(2,q)$, etc. until counter (α,q) is empty.

The sequence of instructions of M_x in every accepting computation is therefore in $d_1^* \cdots d_l^*$, where the sequence $d_1, \ldots, d_l$ is obtained from $a_1, \ldots, a_n$ by replacing each occurrence of C_x at position $i_p, 1 \leq p \leq \alpha$ by $C_{(p,1)}, \ldots, C_{(p,\beta)}$, and replacing each occurrence of D_x at position $j_q, 1 \leq q \leq \beta$ by $D_{(1,q)}, \ldots, D_{(\alpha,q)}$.

Let $w \in L(M)$ and consider an accepting computation on w. Then consider counter x with $f(x)$ as in Equation (1) and $g(x)$ as in Equation (2). Let γ_{i_p} be the number of times that counter x is increased in section i_p, for $1 \leq p \leq \alpha$, in the accepting computation, and let θ_{j_q} be the number of times that counter x is decreased in section j_q, for $1 \leq q \leq \beta$ in the accepting computation. Since M is well-formed, $\gamma_{i_1} + \cdots + \gamma_{i_\alpha} = \theta_{j_1} + \cdots + \theta_{j_\beta}$. Then w can be accepted in M_x as follows: for each section i_p, for $1 \leq p \leq \alpha$, M_x adds to counters $(p,1), \ldots, (p,\beta)$ by amounts $\gamma_{(p,1)}, \ldots, \gamma_{(p,\beta)}$ respectively (these amounts determined in the algorithm below), and for each section j_q, for $1 \leq j \leq \beta$, M_x subtracts from counter $(1,q), \ldots, (\alpha,q)$ by amounts $\theta_{(1,q)}, \ldots, \theta_{(\alpha,q)}$ respectively, such that, for each $1 \leq p \leq \alpha, 1 \leq q \leq \beta$, the following are true:

$$\begin{aligned} \gamma_{i_p} &= \gamma_{(p,1)} + \cdots + \gamma_{(p,\beta)}, && (3) \\ \theta_{j_q} &= \theta_{(1,q)} + \cdots + \theta_{(\alpha,q)}, && (4) \\ \gamma_{(p,q)} &= \theta_{(p,q)}. \end{aligned}$$

If Equations (3) and (4) are true, then the simulation increases and decreases the same amount as the computation of M and can therefore accept in M_x, but where each new counter is increased and decreased in exactly one section.

Intuitively, the situation can be visualized as follows:

$$\begin{array}{ccc|ccc} \gamma_{i_1}, & \ldots, & \gamma_{i_\alpha} & \theta_{j_1}, & \ldots, & \theta_{j_\beta} \\ \hline \gamma_{(1,1)} & & \gamma_{(\alpha,1)} & \gamma_{(1,1)} & \cdots & \gamma_{(1,\beta)} \\ \vdots & & \vdots & & & \\ \gamma_{(1,\beta)} & & \gamma_{(\alpha,\beta)} & \gamma_{(\alpha,1)} & \cdots & \gamma_{(\alpha,\beta)} \end{array}$$

These amounts can be simulated in a "greedy" fashion, by the following algorithm where $\gamma_{(p,q)}$ is the output, for all $1 \leq p \leq \alpha, 1 \leq q \leq \beta$:

```
let X(p) = γ_{i_p}, 1 ≤ p ≤ α; let Y(q) = θ_{j_q}, 1 ≤ q ≤ β;
let p = 1; q = 1; γ_{(p',q')} = 0, ∀ p', q', 1 ≤ p' ≤ α, 1 ≤ q' ≤ β;
while (p ≤ α and q ≤ β)
    γ_{(p,q)} = min{X(p), Y(q)};
    X(p) = X(p) − γ_{(p,q)};
    Y(q) = Y(q) − γ_{(p,q)};
    if (X(p) = 0) then p++;
    if (Y(q) = 0) then q++;
```

In this algorithm, X and Y are initialized to hold γ_{i_p} and θ_{j_q}, for each $1 \leq p \leq \alpha, 1 \leq q \leq \beta$. And, as amounts from each are added to various "counters" $\gamma_{(p,q)}$ in line 4, these same amounts are simultaneously reduced from $X(p)$ and $Y(q)$ in lines 5 and 6 until they are zero.

We will show by induction that each time line 3 is executed, $X(p) = \gamma_{i_p} - \gamma_{(p,q-1)} - \cdots - \gamma_{(p,1)}$ and $Y(q) = \theta_{j_q} - \gamma_{(p-1,q)} - \cdots - \gamma_{(1,q)}$, and after line 6 is executed, $X(p) = \gamma_{i_p} - \gamma_{(p,q)} - \cdots - \gamma_{(p,1)}$ and $Y(q) = \theta_{j_q} - \gamma_{(p,q)} - \cdots - \gamma_{(1,q)}$.

The base case, when $p = 1 = q$ the first time line 3 is executed is true because $X(1) = \gamma_{i_1}$ and $Y(1) = \theta_{j_1}$.

Assume it is true at some iteration when reaching line 3, with $p \leq \alpha$ and $q \leq \beta$. Then $\gamma_{(p,q)} = \min\{\gamma_{i_p} - \gamma_{(p,q-1)} - \cdots - \gamma_{(p,1)}, \theta_{j_q} - \gamma_{(p-1,q)} - \cdots - \gamma_{(1,q)}\}$ in line 4. Assume the first is minimal. Then, in line 4, $X(p) = \gamma_{(p,q)} = \gamma_{i_p} - \gamma_{(p,q-1)} - \cdots - \gamma_{(p,1)}$, thus $X(p)$ is now zero after line 5, and indeed $0 = \gamma_{i_p} - \gamma_{(p,q)} - \cdots - \gamma_{(p,1)}$, which is what we want by induction. Also, after line 6, $Y(q) = \theta_{j_q} - \gamma_{(p,q)} - \cdots - \gamma_{(1,q)}$. Then, p is increased in line 7, and when reaching line 3, $X(p)$ (where p has been increased) is equal to γ_{i_p} which is what we want since $\gamma_{(p,q-1)}, \ldots, \gamma_{(p,1)}$ are all zero. Furthermore, $Y(q) = \theta_{j_q} - \gamma_{(p-1,q)} - \cdots - \gamma_{(1,q)}$ since p was increased from line 6 of the previous iteration. Similarly when the second case is minimal.

Since $\gamma_{i_1} + \cdots + \gamma_{i_\alpha} = \theta_{j_1} + \cdots + \theta_{j_\beta}$, and the same values are subtracted from some $X(p)$ and $Y(q)$ at each step, all $X(p)$ and $Y(q)$ must decrease to 0, and the final iteration has to occur when $p = \alpha, q = \beta$, and in this case $X(p) = Y(q)$. Further, for each p, q in the loop where $X(p)$ is set to 0, γ_{i_p} is the sum of $\gamma_{(p,q)}, \ldots, \gamma_{(p,1)}$, and since p is increased and never decreased again, $\gamma_{(p,q+1)}, \ldots, \gamma_{(p,\beta)}$ are always 0. Thus, Equation (3) is true. Similarly when $Y(q)$ hits zero demonstrates that Equation (4) is true. Thus, since M_x simulates M with these values $\gamma_{(p,q)}$ calculated nondeterministically, they can be set to the amounts calculated by this algorithm. Hence, $w \in L(M_x)$.

Let $w \in L(M_x)$. Then, for counter x, the number of times counter (p, q) is increased is equal to the number of times it is decreased. Furthermore, M_x increases counters $(p, 1), \ldots, (p, \beta)$ consecutively, for each p, $1 \leq p \leq \alpha$, which can be simulated in M by the construction, by increasing counter x this many times in this section. Similarly, M_x decreases from counters $(1, q), \ldots, (\alpha, q)$ consecutively, for each q, $1 \leq q \leq \beta$ in the qth section, which can be simulated by M by the construction, by decreasing counter x this many times. Furthermore, the sum that counter x increases is the same as the amount that it decreases. Thus, $w \in L(M)$.

Thus, $L(M) = L(M_x)$. Continuing this procedure inductively for every counter y where either C_y or D_y occurs more than once yields the result.

It is clear that the procedure works works identically for $\mathsf{LB}_i\mathsf{LB}_d$. For LB_d, it is simpler, since there is no restrictions on the increasing instructions, and similarly for LB_i.

For $\mathsf{LB}_i\mathsf{BD}_d$, the process is similar but an extra step is involved. First, letters that repeat multiple times in the letter-bounded increasing sections are eliminated one at a time according to a similar procedure. For example, say the instruction language of M is a subset of the language $C_1^* C_2^* C_1^* C_3^* C_2^* (D_1 D_2 D_1)^* (D_2 D_3)^* (D_1 D_3)^*$. First, multiple copies of C_1 say are eliminated by introducing new counters $(1, 1), (1, 2), (2, 1), (2, 2)$ as above, where the first coordinate is over the number of occurrences of C_1 in the increasing section, and the second coordinate is over the number of words containing D_1 in the decreasing section. Then, M is simulated by M_1 using the procedure above, where instead of increasing according to pattern C_1 in the first section, M_1 increases counter $(1, 1)$, then nondeterministically switches to $(1, 2)$. Then when simulating the second section of C_1, M_1 increases counter $(2, 1)$ then switches to $(2, 2)$. Therefore, M_1 is increasing according to the pattern $C_{(1,1)}^* C_{(1,2)}^* C_2^* C_{(2,1)}^* C_{(2,2)}^* C_3^* C_2^*$. In the decreasing section, instead of decreasing according to pattern $(D_1 D_2 D_1)^* (D_2 D_3)^* (D_1 D_3)^*$, M_1 decreases according to pattern

$(D_{(1,1)}D_2D_{(1,1)})^*(D_{(2,1)}D_2D_{(2,1)})^*(D_2D_3)^*(D_{(1,2)}D_3)^*(D_{(2,2)}D_3)^*$. Essentially, M_1 is nondeterministically guessing how much of counter 1 will be decreased in the various bounded sections.

Then, after re-numbering these new counters to be consecutive numbers, and repeating this procedure for every counter where some C_x occurs multiple times, results in an instruction language where each C_i occurs exactly once (although running this procedure amplifies the number of each D_i symbols within the bounded words).

To remove multiple copies of each letter D_x that occurs multiple times within the words, consider a machine M which satisfies $I = C_{i_1}^* \cdots C_{i_k}^* w_1^* \cdots w_m^*$, where each element of $\Delta_{(k,c)}$ occurs exactly once in $C_{i_1}, \ldots, C_{i_k}$, $w_i \in \Delta_{(k,d)}^+$, $1 \leq i \leq m$ and some D_x occurs multiple times in $w_1, \ldots, w_m$. We will eliminate multiple copies of D_x for each counter x, one at a time. Then we create M_x as follows: if D_x occurs $\beta > 1$ times (where D_x occurring multiple times within a single word counts as multiple occurrences) within the words $w_1, \ldots, w_m$, then introduce counters $(x,1), \ldots, (x,\beta)$. Instead of increasing from counter x (which only happens in one section), M_x increases from counter $(x,1)$, then nondeterministically switches to $(x,2)$, etc. until counter (x,β). Intuitively, M_x is guessing how much of counter x will be decreased by the pth occurrence of D_x in $w_1, \ldots, w_m$. Then, when simulating the decrease of counter x, M_x decreases according to the pattern $w_1', \ldots, w_m'$, where each w_i' is obtained from w_i by replacing the pth occurrence of D_x with $D_{(x,p)}$. (M_x must remember the words $w_1', \ldots, w_m'$ in the finite control and keep track of which word w_i' and the position within w_i' it is currently simulating in order to decrease the counters in the appropriate order.) For example, if M is decreasing according to $w_1 = (D_1D_1)^*$, then when increasing according to the pattern C_1^*, M_1 uses counter $(1,1)$, then switches to $(1,2)$. Then when decreasing, M_x decreases according to the pattern $(D_{(1,1)}D_{(1,2)})^*$ (which it can do by remembering $(D_{(1,1)}D_{(1,2)})$ in the finite control). Thus, during the increase, M_1 is guessing how much will be consumed by the first and second occurrence of D_1 and then decreasing the appropriate counters. Hence, $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d)) = \mathcal{L}(\mathsf{NCM}(\underline{\mathsf{LB}}_i\underline{\mathsf{BD}}_d))$.

Similarly with $\mathsf{BD}_i\mathsf{LB}_d$. $\square$

The next goal is to separate some families of NCM languages with different instruction languages.

A (quite technical) lemma that is akin to a pumping lemma is proven, but is done entirely on derivations rather than words, so that it can be used twice starting from the same derivation within Proposition 5.

First, we require the following definition. Given an NCM machine M, a derivation of M, $(p_0, w_0, c_{0,1}, \ldots, c_{0,k}) \vdash_M^{t_1} \cdots \vdash_M^{t_m} (p_m, w_m, c_{m,1}, \ldots, c_{m,k})$, is called *collapsible*, if there exists $i, j, 0 \leq i < j \leq m$ such that $p_i = p_j, w_i = w_j$, and $c_{i,l} = c_{j,l}$, for all l, $1 \leq l \leq k$, and non-collapsible otherwise. It is clear that given any accepting computation, there is another that can be constructed that is non-collapsible, simply by eliminating configurations from the original.

Lemma 2. *Let $M = (k, Q, \Sigma, \lhd, \delta, q_0, F)$ be a well-formed k-counter machine in $\mathsf{NCM}(\mathsf{LB}_{id})$ over a distinct-letter-bounded instruction language. Consider any non-collapsible accepting derivation*

$$(p_0, w_0, c_{0,1}, \ldots, c_{0,k}) \vdash_M^{t_1} \cdots \vdash_M^{t_m} (p_m, w_m, c_{m,1}, \ldots, c_{m,k}),$$

where $p_0 = q_0, c_{0,j} = c_{m,j} = 0, 1 \leq j \leq k, p_m \in F, w_m = \lambda$. Assume that there exists x, y, $0 < x \leq y \leq m$ such that $p_{x-1} = p_y$, and this state occurs at least $|Q| + 2$ times in $p_{x-1}, p_x, \ldots, p_y$, and $|h_\Sigma(t_x \cdots t_y)| > 0$, $h_\Delta(t_x \cdots t_y) \in C_i^* \cup D_i^*$, for some i, $1 \leq i \leq k$. Then at least one of the following are true:

1. *there exists r, s with $x \leq r \leq s \leq y$ and an accepting derivation on transition sequence $t_1t_2 \cdots t_{r-1}(t_r \cdots t_s)^2 t_{s+1} \cdots t_m$, with $|h_\Sigma(t_r \cdots t_s)| > 0$,*

2. $h_\Delta(t_x \cdots t_y) \in C_i^+$ *and there exists* r, s *with* $x \leq y \leq r \leq s$ *and* $k_1, k_2 > 1$ *such that the sequence*

$$t_1 t_2 \cdots t_{x-1}(t_x t_{x+1} \cdots t_y)^{k_1} t_{y+1} \cdots t_{r-1}(t_r t_{r+1} \cdots t_s)^{k_2} t_{s+1} \cdots t_m,$$

is an accepting computation, and $h_\Delta(t_r \cdots t_s) \in D_i^+$,

3. $h_\Delta(t_x \cdots t_y) \in D_i^+$ *and there exists* r, s *with* $r \leq s \leq x \leq y$ *and* $k_1, k_2 > 1$ *such that the sequence*

$$t_1 t_2 \ldots t_{r-1}(t_r t_{r+1} \cdots t_s)^{k_1} t_{s+1} \cdots t_{x-1}(t_x t_{x+1} \cdots t_y)^{k_2} t_{y+1} \cdots t_m,$$

is an accepting computation, and $h_\Delta(t_r \cdots t_s) \in C_i^+$.

Proof. Consider a derivation as above, satisfying the stated assumptions. Let $q = p_{x-1} = p_y$ be the state, and i the counter.

Between any two consecutive occurrences of q in the subderivation $t_x, \ldots, t_y$, if counter i does not change, then at least one input letter must get read since this derivation is non-collapsible (and since only counter i can change in this sequence). Furthermore, repeating this sequence of transitions between q and itself twice must be an accepting computation, since the state repeats, the counters have not changed, and at least one extra input letter is read. In this case, 1) is true.

Otherwise, between every two consecutive occurrences of q in the subderivation $t_x, \ldots, t_y$, counter i must change (and either an input letter is read, or not). Thus, between the first and last occurrence of q in $t_x, \ldots, t_y$, at least one input letter is read (by assumption), and the counter must change at least $|Q| + 1$ times (since q occurs $|Q| + 2$ times in the sequence), either increasing if $h_\Delta(t_x \cdots t_y) \in C_i^+$, or decreasing if $h_\Delta(t_x \cdots t_y) \in D_i^+$.

For the first case, assume $h_\Delta(t_x \cdots t_y) \in C_i^+$. Within $t_x \cdots t_y$, counter i increases by z say, where $z > |Q|$. Hence, counter i must decrease by z as well since M is well-formed. Since the instruction language is in LB_{id}, all the decreasing of counter i must be within the derivation where no other counter is changed. When counter i decreases, there must also be a state p that appears twice with at least one decrease in between this repeated state since $z > |Q|$. Let $z' > 0$ be the amount the counter is decreased between p and itself. That is, there must exist $r \leq s$ such that $p_{r-1} = p = p_s$ and counter i is decreased by $z' > 0$ within this part of the derivation.

Then, create a derivation from the derivation above where, during the cycle that increases counter i by z, we increase by $z + zz'$ (by iterating this cycle $1 + z' = k_1 > 1$ times), and during the cycle that decreases counter i by z', we instead decrease the counter by $z' + zz'$ (by iterating this cycle $1 + z = k_2 > 1$ times). This new computation must accept and 2) is true.

The case is similar if $h_\Delta(t_x \cdots t_y) \in D_i^+$, with 3) being true. □

The next result follows from Lemma 1 and this new pumping lemma.

Proposition 4. $\{a^n b^n c^n \mid n > 0\} \notin \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$.

Proof. Assume otherwise. Let L be the language in the statement, and let M be a well-formed k-counter machine accepting L, over a distinct-letter-bounded instruction alphabet where each character of Δ_k occurs exactly once, $I \subseteq a_1^* \cdots a_{2k}^*$, which is enough by Lemma 1. Let Q be the state set of M.

Let $n = (|Q| + 1)(|Q| + 2)(2k + 1)$. Then, on input $w = a^n b^n c^n$, consider a non-collapsible accepting computation on transition sequence $t_1 t_2 \cdots t_m$ of M; that is,

$$(p_0, w_0, c_{0,1}, \ldots, c_{0,k}) \vdash_M^{t_1} \cdots \vdash_M^{t_m} (p_m, w_m, c_{m,1}, \ldots, c_{m,k}),$$

where $p_0 = q_0, c_{0,j} = 0 = c_{m,j}, 1 \leq j \leq k, p_m \in F, w_m = \lambda$, and $w = w_0$.

Then, when reading the a's there must exist x', y', $0 < x' \leq y' \leq m$ such that $h_\Delta(t_{x'} \cdots t_{y'}) \in \{C_i, D_i\}^*$, for some i, $1 \leq i \leq 2k$, such that $|h_\Sigma(t_{x'} \cdots t_{y'})| \geq (|Q|+1)(|Q|+2)$ (at least $(|Q|+1)(|Q|+2)$ a's are read while increasing or decreasing counter i). Then, at least this many transitions are applied during this sequence of transitions. Then, some state q occurs at least $|Q|+2$ times in this subderivation, with at least one input letter read between the first and last occurrence of q. Hence, Lemma 2 must apply.

If case 1 is true, this produces a word with more a's than b's.

If case 2 is true, then (using the variables in the Lemma 2 statement), this derivation has more than n a's since $k_1 > 1$ and $|h_\Sigma(t_x \cdots t_y)| > 0$, and therefore $h_\Sigma(t_r \cdots t_s)$ would need to consist of both b's and c's, otherwise words would be produced with more b's than c's, or more c's than b's. But then, there are words that are not in $a^*b^*c^*$, a contradiction.

If case 3 is true, then this produces a word with more a's than b's and c's. $\square$

In addition, the following can be shown with Lemma 1 and two applications of the pumping lemma.

Proposition 5. $\{a^n b^n c^l d^l \mid n, l > 0\} \notin \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i \mathsf{LB}_d))$.

Proof. Assume otherwise. Let L be the language in the statement, and let M be a well-formed k-counter machine accepting L, over a distinct-letter-bounded instruction alphabet where each letter of Δ_k occurs exactly once, $I \subseteq a_1^* \cdots a_{2k}^*$, which is enough by Lemma 1. Let Q be the state set of M.

Let $n = (|Q|+1)(|Q|+2)(2k+1)$. Then, on input $w = a^n b^n c^n d^n$, consider a non-collapsible accepting computation on $t_1 t_2 \cdots t_m$ of M accepting w; that is,

$$(p_0, w_0, c_{0,1}, \ldots, c_{0,k}) \vdash_M^{t_1} \cdots \vdash_M^{t_m} (p_m, w_m, c_{m,1}, \ldots, c_{m,k}),$$

where $p_0 = q_0, c_{0,j} = 0 = c_{m,j}, 1 \leq j \leq k, p_m \in F, w_m = \lambda$, and $w = w_0$.

Then, when reading the a's there must exist x', y', $0 < x' \leq y' \leq m$ such that $h_\Delta(t_{x'} \cdots t_{y'}) \in \{C_i, D_i\}^*$, for some i, $1 \leq i \leq 2k$, such that $|h_\Sigma(t_{x'} \cdots t_{y'})| \geq (|Q|+1)(|Q|+2)$ (at least $(|Q|+1)(|Q|+2)$ a's are read while increasing or decreasing counter i). Then, at least this many transitions are applied during this sequence of transitions. Then, some state q occurs at least $|Q|+2$ times in this subderivation, with at least one input letter read between the first and last occurrence of q. Hence, Lemma 2 must apply.

If case 1 is true, this produces a word with more a's than b's.

If case 3 is true, then this produces a word with more a's than b's.

Assume for the rest of this proof then that case 2 is true. Then, there exists r, s with $x \leq y \leq r \leq s$ and $k_1, k_2 > 1$ such that the sequence

$$t_1 t_2 \cdots t_{x-1} (t_x t_{x+1} \cdots t_y)^{k_1} t_{y+1} \cdots t_{r-1} (t_r t_{r+1} \cdots t_s)^{k_2} t_{s+1} \cdots t_m,$$

is an accepting computation, and $h_\Delta(t_r \cdots t_s) \in D_i^+$. The word accepted, has more than n a's, n' say, since $k_1 > 1$ and $|h_\Sigma(t_x \cdots t_y)| > 0$, and the only way to not obtain a contradiction would be if $h_\Sigma(t_r \cdots t_s)$ consists of only b's such that the resulting input word reading during this derivation also has n' b's. Also, because $h_\Delta(t_r \cdots t_s) \in D_i^+$, it follows that a counter has already started decreasing while reading the b's. Therefore, after this point of the derivation, no counter can increase again since $M \in \mathsf{NCM}(\mathsf{LB}_i \mathsf{LB}_d)$.

Although this new derivation is potentially collapsible (since new transitions were added in from $t_1 \cdots t_m$), as mentioned earlier, it is possible to obtain a non-collapsible derivation from this new derivation simply by removing configurations (entirely in the new sections added while reading

a's and b's). Then, a new derivation can be obtained on transition sequence $s_1 \cdots s_{m'}$ accepting $a^{n'} b^{n'} c^n d^n$.

Then consider this non-collapsible accepting derivation and consider the subsequence when reading the c's. There must exist x'', y'', $0 < x'' \leq y'' \leq m'$ such that $h_\Delta(t_{x''} \cdots t_{y''}) \in \{D_j\}^*$, for some i, $1 \leq j \leq k$, such that $|h_\Sigma(t_{x''} \cdots t_{y''})| \geq (|Q|+1)(|Q|+2)$ (it must be D_j since this derivation has already started decreasing while reading the b's). Then, at least this many transitions are applied during this sequence of transitions. Then, some state q' occurs at least $|Q|+2$ times in this sub-derivation, with at least one input letter read between the first and last occurrence of q'. Hence, Lemma 2 must again apply.

If case 1 applies, then this produces a word with more c's than d's, as does case 3, and case 2 cannot apply since the derivation has already started decreasing. Thus, we obtain a contradiction. □

Therefore, the following is immediate:

Proposition 6. $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d)) \subsetneq \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id})) \subsetneq \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$.

Proof. The inclusions follow from the definitions. Strictness of the first inclusion follows from Proposition 5, as $\{a^n b^n c^l d^l \mid n, l > 0\} \in \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$ by making a 2-counter machine that reads a's and adds the number of a's to the first counter, then reads b's while verifying that this number is the same, and then reads c's while adding the number of c's to the second counter, then reads d's while decreasing the second counter, verifying that the number is the same.

Strictness of the second inclusion follows by Proposition 4 as $\{a^n b^n c^n \mid n > 0\} \in \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$ by building a 2-counter machine that adds 1 to counter 1 then 2 repeatedly for each a read, then verifies that the contents of the first counter is the same as the number of b's, then verifies that the contents of the second counter is the same as the number of c's. □

4 Generators for the Families

We will go through certain families individually while creating a more restricted set of generators than is provided by Proposition 3.

First, we will give two characterizations of $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$.

Proposition 7. *$\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$ is the smallest full trio containing all distinct-letter-bounded languages of the form $\{a_1^{i_1} \cdots a_m^{i_m} \mid a_j = C_l, a_n = D_l \text{ imply } i_j = i_n\}$, where $a_1, \ldots, a_m$ is a permutation of Δ_k such that $a_j = C_l, a_n = D_l$ implies $j < n$.*

Proof. It follows from Lemma 1 that every language in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$ can be obtained by an instruction language in $\mathcal{I}$, where $\mathcal{I}$ is the distinct-letter-bounded subset of LB_{id}. Thus, $\mathcal{L}(\mathsf{NCM}(\mathcal{I})) = \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$. From Proposition 2, it follows that $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ is the smallest full trio containing $\mathcal{I}_{eq}$. Furthermore, $\mathcal{I}_{eq}$ is equal to the languages in the proposition statement. □

A similar characterization can be obtained with a single language for each k.

Proposition 8. *Let $k \geq 1$, and let $L_k^{\mathsf{LB}_{id}} = \{a_1^{i_1} a_2^{i_2} \cdots a_m^{i_m} \mid \{a_1, \ldots, a_m\}$ is a permutation of Δ_k, and $(C_j = a_l, D_j = a_n$ implies both $l < n$ and $i_l = i_n)$, for each $j, 1 \leq j \leq k\}$.*

Then $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$ is the smallest full trio containing $L_k^{\mathsf{LB}_{id}}$, for each $k \geq 1$.

Proof. It is clear that $L_k^{\mathsf{LB}_{id}}$ is the finite union of languages of the form of Proposition 7, and since this family is closed under union by Proposition 3, then $L_k^{\mathsf{LB}_{id}} \in \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$. Further, all bounded languages I of the form of Proposition 7 can be obtained by intersecting $L_k^{\mathsf{LB}_{id}}$ with the regular language $a_1^* \cdots a_m^*$. $\square$

Next, we will give characterizations for $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$, whose proof is similar to Proposition 7.

Proposition 9. $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$ *is the smallest full trio containing all distinct-letter-bounded languages of the form* $\{a_1^{l_1} \cdots a_k^{l_k} b_1^{j_1} \cdots b_k^{j_k} \mid a_i = C_m, b_n = D_m$ *imply* $l_i = j_n\}$, *where* $a_1, \ldots, a_k$ *is a permutation of* $\Delta_{(k,c)}$ *and* $b_1, \ldots, b_k$ *is a permutation of* $\Delta_{(k,d)}$.

This can similarly be turned into one language for each k, as follows with a proof similar to Proposition 8:

Proposition 10. *Let* $k \geq 1$, *and let* $L_k^{\mathsf{LB}_i\mathsf{LB}_d} = \{a_1^{l_1} \cdots a_k^{l_k} b_1^{j_1} \cdots b_k^{j_k} \mid a_1, \ldots, a_k$ *is a permutation of* $\Delta_{(k,c)}$, $b_1, \ldots, b_k$ *is a permutation of* $\Delta_{(k,d)}$, *and* ($C_m = a_i, D_m = b_n$ *implies* $l_i = j_n$), *for each* $j, 1 \leq j \leq k\}$.

Then $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$ *is the smallest full trio containing* $L_k^{\mathsf{LB}_i\mathsf{LB}_d}$, *for each* $k \geq 1$.

Next, we will provide an alternate interesting characterization for both families using properties of semilinear sets. Let $m \geq 1$. A linear set $Q \subseteq \mathbb{N}_0^n, n \geq 1$, is m-bounded if the periodic vectors of Q have at most m non-zero coordinates. (There is no restriction on the constant vector.) A semilinear set Q is m-bounded if it is a finite union of m-bounded linear sets.

Let $L \subseteq a_1^* \cdots a_n^*, a_1, \ldots, a_n \in \Sigma$ be a distinct-letter-bounded language. L is called a *distinct-letter-bounded 2-bounded semilinear language* if there exists a 2-bounded semilinear set Q such that $L = \{a_1^{i_1} \cdots a_n^{i_n} \mid (i_1, \ldots, i_n) \in Q\}$. L is called a *distinct-letter-bounded 2-bounded overlapped semilinear language* if there exists a 2-bounded semilinear set Q with the property that in any of the linear sets comprising Q, there are no periodic vectors v with non-zero coordinates at positions $i < j$, and v' with non-zero coordinates at positions $i' < j'$ such that $1 \leq i < j < i' < j' \leq n$, and $L = \{a_1^{i_1} \cdots a_n^{i_n} \mid (i_1, \ldots, i_n) \in Q\}$. (They overlap in the sense that, for any such Q, v, i, j, v', i', j', then the interval $[i, j]$ must overlap with $[i', j']$.)

As above, we can also define distinct-letter-bounded 1-bounded semilinear languages. Clearly, these languages are regular and, hence, contained in any nonempty full trio family [2]. For 2-bounded, the following is true:

Proposition 11.

1. $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$ *is the smallest full trio containing all distinct-letter-bounded 2-bounded semilinear languages.*
2. $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$ *is the smallest full trio containing all distinct-letter-bounded 2-bounded overlapped semilinear languages.*

Proof. For Part 1, let $Q \subseteq \mathbb{N}_0^n$ be a 2-bounded semilinear set. It will be shown that $L = \{a_1^{i_1} \cdots a_n^{i_n} \mid (i_1, \ldots, i_n) \in Q\}$ is accepted by an $M \in \mathsf{NCM}(\mathsf{LB}_{id})$. It is sufficient prove the case when Q is a linear set by Proposition 3. For ease in notation, we illustrate the construction of M with an example, which is easy to generalize. Let

$$Q = \{(5,4,2,0,0,3) + i(1,0,3,0,0,0) + j(2,3,0,0,0,0) + k(0,4,0,2,0,0) \\ + m(1,0,6,0,0,0) + n(0,0,2,0,7,0) + r(0,0,0,0,2,0) + s(0,0,0,0,0,8) \mid i,j,k,m,n,r,s \geq 0\}.$$

Then $L = \{a^{5+i+2j+m} b^{4+3j+4k} c^{2+3i+6m+2n} d^{0+2k} e^{0+7n+2r} f^{3+8s} \mid i,j,k,m,n,r,s \geq 0\}$.

Let $a_0, a_1, a_2, a_3, b_0, b_1, b_2, c_0, c_1, c_2, c_3, d_0, d_1, e_0, e_1, e_2, f_0, f_1$ be distinct symbols. Let

$$L' = \{a_0^5 a_1^i a_2^{2j} a_3^m b_0^4 b_1^{3j} b_2^{4k} c_0^2 c_1^{3i} c_2^{6m} c_3^{2n} d_0^0 d_1^{2k} e^0 e_1^{7n} e_2^{2r} f_0^3 f_1^{8s} \mid i, j, k, m, n, r, s \geq 0\}.$$

Thus $L' \subseteq a_0^* a_1^* a_2^* a_3^* b_0^* b_1^* b_2^* c_0^* c_1^* c_2^* c_3^* d_0^* d_1^* e_0^* e_1^* e_2^* f_0^* f_1^*$. M' will have counters C_i, C_j, C_m, C_k, C_n. It is straightforward to construct M' to accept L' whose counter behavior is contained in the language $C_i^* C_j^* C_m^* D_j^* C_k^* D_i^* D_m^* C_n^* D_k^* D_n^*$. (Note that M' does not need counters to check the constants and to check the $2r$ and $8s$ portions. As usual, we assume that for any symbol x, $x^0 = \lambda$.)

Let h be a homomorphism which maps a_0, a_1, a_2, a_3 to a; b_0, b_1, b_2 to b; c_0, c_1, c_2, c_3 to c; d_0, d_1 to d; e_0, e_1, e_2 to e; f_0, f_1 to f. Then $L = h(L')$.

The construction above is easy to generalize. For each position p (e.g., position 1), we split the symbol in that position (e.g., a) into 1 plus the number of periodic vectors with non-zero values in position p (e.g., a is split into split- symbols a_0, a_1, a_2, a_3). Then these symbols are assigned exponents that represent the number of times the corresponding non-zero values need to be repeated (e.g., $a_0^5, a_1^{1\times i}, a_2^{2\times j}, a_3^{1\times m}$, where 5 is the non-zero value at position 1 in the constant vector, and 1, 2, 1 are the non-zero values at position 1 in the periodic vectors with parameters i, j, m representing the number of times the corresponding non-zero values have to be repeated). Then a counter is assigned to a split-symbol at position p if and only if there there is a periodic vector with non-zero values at positions p and q with $p < q$. (For example, counter C_i is assigned to symbol a_1, C_j is assigned to a_2, C_m is assigned to a_m, C_k is assigned to b_2, C_n is assigned to c_3. However, no counters are assigned to $a_0, b_0, c_0, d_0, e_0, f_0$ because they correspond to the constant vector, and no counters are assigned to e_2 and f_1.) Then one can easily determine the counter behavior pattern which would guide the computation of the $\mathsf{NCM}(\mathsf{LB}_{id})$ machine.

Conversely, $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$ is the smallest full trio containing all distinct-letter-bounded languages of the form of Proposition 9, by that proposition. Further, all of these are distinct-letter-bounded 2-bounded semilinear languages. Thus, Part 1 follows.

The proof for Part 2 is similar to the above proof. $\square$

Next we will give a characterization of the smallest full trio containing all bounded $\mathcal{L}(\mathsf{CFL})$ languages. For that, we consider instruction family StLB_{id}. An example of an StLB_{id} language (counter behavior) is $C_1^* C_2^* C_3^* D_3^* C_2^* D_2^* C_1^* D_1^*$. But the counter behavior $C_1^* C_2^* C_3^* D_3^* C_2^* D_2^* C_1^* D_2^* C_1^* D_1^*$ is not an StLB_{id} language since C_2 appears, then C_1, then D_2, then D_1, violating the StLB_{id} definition.

The next results show that $\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id}))$ is the smallest full trio containing all bounded context-free languages. It has previously been found that there is no principal full trio (ie. generated by a single language [2]) accepting these languages [11] (this paper does not use the 'principal' notation). Our proof uses a known characterization of distinct-letter-bounded context-free languages (CFLs) from [3].

Proposition 12. $\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id}))$ *is the smallest full trio containing all bounded context-free languages.*

Proof. First we show containment. For every bounded $L \in \mathcal{L}(\mathsf{CFL}), L \subseteq w_1^* \cdots w_n^*$, for distinct $a_1, \ldots, a_n$, then $L' = \{a_1^{i_1} \cdots a_n^{i_n} \mid w_1^{i_1} \cdots w_n^{i_n} \in L\}$ must also be in $\mathcal{L}(\mathsf{CFL})$ by using a finite transducer that reads w_i and outputs a_i, as $\mathcal{L}(\mathsf{CFL})$ is closed under finite transductions [2]. Then since $\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id}))$ is a full trio, it is sufficient to show that every distinct-letter-bounded $\mathcal{L}(\mathsf{CFL})$ $L' \subseteq a_1^* \cdots a_n^*$ can be accepted by an $\mathsf{NCM}(\mathsf{StLB}_{id})$, as $L = h(L')$ for a homomorphism h that outputs w_i from a_i.

Assume that $n \geq 2$. (The case $n = 1$ is trivial, since L is regular.)

We will prove the claim by induction on n. The result holds for $n = 2$, since it is known that for every $\mathcal{L}(\mathsf{CFL})$ $L \subseteq w_1^* w_2^*$ (where $w_1, w_2 \in \Sigma^+$), L can be accepted by an $\mathsf{NCM}(1, 1)$, hence by an $\mathsf{NCM}(\mathsf{StLB}_{id})$ [8].

Now suppose $n \geq 3$. The following characterization is known [3]. For all $\Sigma = \{a_1, \ldots, a_n\}$, $n \geq 3$, then each $\mathcal{L}(\mathsf{CFL})$ $L \subseteq a_1^* \cdots a_n^*$ is a finite union of sets of the following form:

$$M(D, E, F) = \{a_1^i x y a_n^j \mid a_1^i a_n^j \in D, x \in E, y \in F\},$$

where $D \subseteq a_1^* a_n^*$, $E \subseteq a_1^* \cdots a_q^*$, $F \subseteq a_q^* \cdots a_n^*$, $1 < q < n$, are in $\mathcal{L}(\mathsf{CFL})$, and conversely, each finite union of sets of the form $M(D, E, F)$ is a $\mathcal{L}(\mathsf{CFL})$ $L \subseteq a_1^* \cdots a_n^*$.

By this, D can be accepted by an $\mathsf{NCM}(\mathsf{StLB}_{id})$ M_1 with 1 counter. By the induction hypothesis, E and F can be accepted by $\mathsf{NCM}(\mathsf{StLB}_{id})s$ M_2 with k_2 and M_3 with k_3 counters, respectively, since they are over smaller alphabets.

Since $\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id}))$ is closed under union, it is sufficient to build an $\mathsf{NCM}(\mathsf{StLB}_{id})$ M accepting $M(D, E, F)$. Then M has $k_2 + k_3 + 1$ counters. On a given input, M starts by simulating M_1, and while still reading a_1's, it remembers the current state of M_1 in the finite control, and starts simulating M_2. (The point when M starts simulating M_2 is nondeterministically chosen, as long as the input head of M has not gone past the a_1's.) After M_2 accepts, M starts simulating M_3. (Again, the point when M starts simulating M_3 is nondeterministically chosen.) When M_3 accepts, M continues the simulation of M_1 from the state it remembered until the string is accepted.

Conversely, let L be any language accepted by an $\mathsf{NCM}(\mathsf{StLB}_{id})$ with k counters. As usual, all counters are zero on acceptance. We construct an NPDA M' that accepts L. M' has stack alphabet $\{Z_0, C_1, \ldots, C_k\}$, where Z_0 denotes the bottom of the stack which is never altered. The stack containing only Z_0 indicates that it is empty. M' accepts on empty stack and final state. Let $I = a_1^* \cdots a_m^* \in \mathsf{StLB}_{id}$ be a superset of the instructions of M. Then M' simulates M faithfully but uses the stack to simulate the counters as follows (noting that a counter C_i is zero if and only if the stack is empty or the top of the stack is not C_i):

– If M increments counter i, M' pushes C_i on the stack.

– If M decrements counter i, M' pops C_i from the stack.

Note that because I is an StLB_{id} instruction, in any accepting computation, when M decrements counter i, the top of the stack must be C_i.

When M accepts (with all counters zero, corresponding to the stack of M' being empty), M' accepts. $\square$

From this, the following can be determined:

Corollary 3.

1. *$\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id})) \subsetneq \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$.*
2. *$\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id})$ and $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$ are incomparable.*

Proof. Obviously, $\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id})$ is contained in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_{id}))$. To show proper containment, let $L = \{a^i b^j c^i d^j \mid i, j \geq 1\}$. Clearly, L can be accepted by an $\mathsf{NCM}(\mathsf{LB}_{id})$ with counter behavior $C_1^* C_2^* D_1^* D_2^*$, but L is not a context-free language. The result follows since every language in $\mathcal{L}(\mathsf{NCM}(\mathsf{StLB}_{id})$ is a context-free language.

For Part 2, L is in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$ but is not a $\mathcal{L}(\mathsf{CFL})$. Now let $L' = \{a^i b^i c^j d^j \mid i, j \geq 1\}$. L' is a $\mathcal{L}(\mathsf{CFL})$, but this is not in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$ by Proposition 5. $\square$

Next, we will show that all bounded Ginsburg semilinear languages are in two of the language families (and therefore in all larger families).

Lemma 3. *All bounded Ginsburg semilinear languages are in $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d))$, $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d))$.*

Proof. By [7], it is enough to show for all distinct-letter-bounded semilinear languages. And since both families are closed under union, by Proposition 3, and since every semilinear set is the finite union of linear sets, it is enough to show for all distinct-letter-bounded semilinear languages induced by a linear set Q.

Let $Q \subseteq \mathbb{N}_0^k$ be the linear set with constant vector $\vec{v_0}$ and periods $\vec{v_1}, \dots, \vec{v_n}$. We will create a machine M where input is of the form $w = a_1^{i_1} \cdots a_k^{i_k}$, each a_i is distinct, and M accepts w if and only if $(i_1, \dots, i_k) \in Q$. We will start with the case for $\mathsf{BD}_i\mathsf{LB}_d$.

Then M first adds $v_0(i)$ to counter i, for each i from $1 \leq i \leq k$, on λ transitions. Then M does the same for $\vec{v_1}$ arbitrarily many times, then the same for $\vec{v_2}$, etc. This insertion pattern is in the bounded language

$$C_1^* \cdots C_k^* (C_1^{\vec{v_1}(1)} C_2^{\vec{v_1}(2)} \cdots C_k^{\vec{v_1}(k)})^* \cdots (C_1^{\vec{v_n}(1)} C_2^{\vec{v_n}(2)} \cdots C_k^{\vec{v_n}(k)})^*.$$

Then the counter contents are in the linear set without having read any input. Then it verifies that the input is equal to the counter contents one counter at a time, and therefore M accepts the distinct-letter-bounded language induced by Q. So the decreasing pattern is $D_1^* D_2^* \cdots D_k^*$ which is letter-bounded.

For $\mathsf{LB}_i\mathsf{BD}_d$, this pattern is inverted, and M starts by placing i_j in counter j, for $1 \leq j \leq k$, according to the letter-bounded pattern $D_1^* \cdots D_k^*$. Then, M subtracts from the counters with the same pattern that it increased from the counters in the case above, which is in the bounded language (replacing all C's with D's). M accepts if all counters reach zero in this fashion. $\square$

From the definition, it is immediate that if $\mathcal{I} \subseteq \mathcal{I}'$, then $\mathcal{L}(\mathsf{NCM}(\mathcal{I})) \subseteq \mathcal{L}(\mathsf{NCM}(\mathcal{I}'))$. It is clear that all of $\mathsf{LB}_{id}, \mathsf{BD}_i\mathsf{LB}_d, \mathsf{LB}_i\mathsf{BD}_d$ are a subset of BD_{id}. We will show that three of these counter families coincide.

Proposition 13. *$\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_i)) = \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d)) = \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$ is the smallest full trio containing all bounded Ginsburg semilinear languages, and the smallest full trio containing all bounded Parikh semilinear languages.*

Proof. All of the families are full trios by Proposition 1. All must contain all bounded Ginsburg semilinear languages by Lemma 3, and therefore all bounded bounded Parikh semilinear languages [7].

To complete the proof, we will now show that all languages in $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$ can be obtained from the bounded Parikh (which are then bounded Ginsburg) semilinear languages using full trio operations.

Let $\mathcal{I} = \mathsf{BD}_{id}$. By Proposition 2, $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$ is the smallest family of languages containing $\mathcal{I}_{eq} = \{I \mid I = \{w = w_1^* \cdots w_m^* \mid |w|_{C_i} = |w|_{D_i},$ for each $1 \leq i \leq k,$ all C_i's appear before any D_i's$\}$, $w_i \in \Delta_k^*, k \geq 1\}$. But every $I_{eq} \in \mathcal{I}_{eq}$ is a bounded Parikh semilinear language. Thus the statement follows. $\square$

Corollary 4. *For all $\mathcal{I} \in \{\mathsf{BD}_i\mathsf{LB}_d, \mathsf{LB}_i\mathsf{BD}_d, \mathsf{BD}_{id}, \mathsf{LB}_d, \mathsf{LB}_i, \mathsf{LB}_\cup, \mathsf{ALL}\}$, then $\mathcal{L}(\mathsf{NCM}(\mathcal{I}))$ contains all bounded Ginsburg semilinear languages, and all bounded languages in $\mathcal{L}(\mathsf{NCM})$.*

Next, we establish two simple sets of generators for $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$. These languages will therefore be a simple mechanism to show whether or not a full trio $\mathcal{L}$ contains every bounded Ginsburg semilinear language, and therefore has exactly the same bounded languages as NCM, and has all bounded languages contained in any semilinear trio.

Proposition 14. *For $k \geq 1$, let*

$$L_k^{\mathsf{BD}_i\mathsf{LB}_d} = \{w_1^{x_1} \cdots w_m^{x_m} D_1^{y_1} \cdots D_k^{y_k} \mid \begin{array}{l} w_j \in \Delta_{(k,c)}^+, x_j > 0, 1 \leq j \leq m, \\ \textit{for } 1 \leq i \leq k, |w_1 w_2 \cdots w_m|_{C_i} = 1, \\ (C_i \in \text{alph}(w_j) \textit{ implies } y_i = x_j)\}, \end{array}$$

$$L_k^{\mathsf{LB}_i\mathsf{BD}_d} = \{C_1^{y_1} \cdots C_k^{y_k} w_1^{x_1} \cdots w_m^{x_m} \mid \begin{array}{l} w_j \in \Delta_{(k,d)}^+, x_j > 0, 1 \leq j \leq m, \\ \textit{for } 1 \leq i \leq k, |w_1 w_2 \cdots w_m|_{D_i} = 1, \\ (D_i \in \text{alph}(w_j) \textit{ implies } y_i = x_j)\}. \end{array}$$

Then $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d)) = \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d)) = \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$ is the smallest full trio containing $L_k^{\mathsf{BD}_i\mathsf{LB}_d}$, for each $k \geq 1$, and also the smallest full trio containing $L_k^{\mathsf{LB}_i\mathsf{BD}_d}$, for each $k \geq 1$.

Proof. The equality of the families follows from Proposition 13. First, we will consider $L_k^{\mathsf{LB}_i\mathsf{BD}_d}$. To show this language is in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d))$, notice that $L_k^{\mathsf{LB}_i\mathsf{BD}_d} \cap C_1^* \cdots C_k^* w_1^* \cdots w_m^*$ is clearly in this family for fixed words $w_1, \ldots, w_m$, where $|w_1 \cdots w_m|_{D_i} = 1$ for each i. Furthermore, there are a finite number of combinations of such words, and $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d))$ is closed under union by Proposition 3, therefore $L_k^{\mathsf{LB}_i\mathsf{BD}_d}$ is in this family.

From Lemma 1 and Proposition 2, it follows that $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d))$ is the smallest full trio containing $(\underline{\mathsf{LB}}_i\underline{\mathsf{BD}}_d)_{eq}$.

First consider a slight variant of $L_k^{\mathsf{LB}_i\mathsf{BD}_d}$,

$$L_k = \{C_{i_1}^{y_1} \cdots C_{i_k}^{y_k} w_1^{x_1} \cdots w_m^{x_m} \mid \begin{array}{l} w_j \in \Delta_{(k,d)}^+, x_j > 0, 1 \leq j \leq m, \\ C_{i_1}, \ldots, C_{i_k} \text{ is a permutation of } \Delta_{(k,c)}, \\ \text{for } 1 \leq l \leq k, |w_1 w_2 \cdots w_m|_{D_l} = 1, \\ (D_{i_l} \in \text{alph}(w_j) \text{ implies } y_l = x_j)\}. \end{array}$$

Then every language $I \in (\underline{\mathsf{LB}}_i\underline{\mathsf{BD}}_d)_{eq}$ over Δ_k^* for some $k \geq 1$ is equal to

$$\{w = C_{i_1}^{y_1} \cdots C_{i_k}^{y_k} w_1^{x_1} \cdots w_m^{x_m} \mid x_j = y_l \text{ if } D_{i_l} \in \text{alph}(w_j) > 0\},$$

where $C_{i_1}, \ldots, C_{i_k}$ is a permutation of $\Delta_{(k,c)}$, $w_j \in \Delta_{(k,d)}^+$, and each letter of $\Delta_{(k,d)}$ occurs exactly once in $w_1 w_2 \cdots w_m$. In this case, $I = L_k \cap C_{i_1}^+ C_{i_2}^+ \cdots C_{i_k}^+ w_1^+ \cdots w_m^+$. Furthermore, consider the homomorphism that maps C_{i_l} to C_l, and D_{i_l} to D_l. Then,

$$I = h^{-1}(L_k^{\mathsf{BD}_i\mathsf{LB}_d} \cap C_1^+ C_2^+ \cdots C_k^+ h(w_1)^+ \cdots h(w_m)^+).$$

The case is similar for $L_k^{\mathsf{BD}_i\mathsf{LB}_d}$. □

Then, by Proposition 13, Proposition 14, and [7], the following is true:

Proposition 15. *Let $\mathcal{L}$ be a full trio. Then, the following are equivalent:*

- *$\mathcal{L}$ contains all bounded Ginsburg semilinear languages,*
- *$\mathcal{L}$ contains all distinct-letter-bounded Ginsburg semilinear languages,*
- *$\mathcal{L}$ contains all bounded Parikh semilinear languages,*
- $\mathcal{L}(\mathsf{NCM})^{\text{bd}} (= \mathcal{L}(\mathsf{DCM})^{\text{bd}} = \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))^{\text{bd}})$ *is contained in $\mathcal{L}$,*
- $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d)) (= \mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{BD}_d)) = \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id})))$ *is contained in $\mathcal{L}$,*
- *$\mathcal{L}$ contains $L_k^{\mathsf{BD}_i\mathsf{LB}_d}$, for each $k \geq 1$,*

- *$\mathcal{L}$ contains $L_k^{\mathsf{LB}_i\mathsf{BD}_d}$, for each $k \geq 1$.*

Furthermore, if $\mathcal{L}$ is also semilinear, then these conditions are equivalent to $\mathcal{L}^{\rm bd} = \mathcal{L}(\mathsf{NCM})^{\rm bd} = \mathcal{L}(\mathsf{DCM})^{bd}$.

By Proposition 4 and Proposition 15, the following is immediate:

Corollary 5. *$\mathsf{NCM}(\mathsf{LB}_{id})$ and $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i\mathsf{LB}_d))$ do not contain all bounded Ginsburg semilinear languages, or all bounded Parikh semilinear languages.*

There is another simple equivalent form of the family $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$. Let SBD_{id} be the subset of BD_{id} that is the family

$$\{I \mid k \geq 1, I = w_1^* \cdots w_m^*, w_i \in \Delta_k^+, |w_i| \leq 2, 1 \leq i \leq m\}.$$

Proposition 16. *$\mathcal{L}(\mathsf{NCM}(\mathsf{SBD}_{id}))$ contains all bounded Ginsburg semilinear languages. Hence, $\mathcal{L}(\mathsf{NCM}(\mathsf{SBD}_{id})) = \mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id}))$.*

Proof. We need only show that every distinct-letter-bounded Ginsburg language induced by a linear set Q is in $\mathcal{L}(\mathsf{NCM}(\mathsf{SBD}_{id}))$ as this family is clearly closed under union. We illustrate with an example, which can easily be generalized.

Let $Q = \{(2,0,3,1) + (4,2,3,6)i + (3,5,1,0)j + (1,0,0,0)k + (0,7,0,2)m \mid i,j,k,m \geq 0\}$. The distinct-letter-bounded language $L = \{a^{2+4i+3j+k}b^{2i+5j+7m}c^{3+3i+j}d^{1+6i+2m} \mid i,j,k,m \geq 0\}$ is induced by this linear set.

Let $a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3, c_0, c_1, c_2, d_0, d_1, d_2$ be new symbols, and let

$$L' = \{a_0^2 a_1^{4i} a_2^{3j} a_3^k b_0^0 b_1^{2i} b_2^{5j} b_3^{7m} c_0^3 c_1^{3i} c_2^j d_0^1 d_1^{6i} d_2^{2m} \mid i,j,k,m \geq 0\}.$$

L' can be accepted by an NCM M with counters $C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8$ which operates as follows when given input w:

– reads the first input segment and verifies that it is a_0^2.
– reads a_1^{4i} and stores i in (C_1, C_2).
– reads a_2^{3j} and stores j in (C_3, C_4).
– reads the next input segment a_3^k.
– reads the next input segment b_0^0.
– reads the next input segment and verifies that it is b_1^{2i} by decrementing C_1 while incrementing C_5.
– reads the next input segment and verifies that it is b_2^{5j} by decrementing C_3.
– reads b_3^{7m} and stores m in C_6.
– reads the next input segment and verifies that it is c_0^3.
– reads the next input segment and verifies that it is c_1^{3i} by decrementing C_2.
– reads the next input segment and verifies that it is c_2^j. by decrementing C_4.
– reads the next input segment and verifies that it is d_0^1.
– reads the next input segment and verifies that it is d_1^{6i} by decrementing C_5.
– reads the next input segment and verifies that it is d_2^{2m} by decrementing C_6.

Then M satisfies $(C_1C_2)^*(C_3C_4)^*(D_1C_5)^*D_3^*C_6^*D_2^*D_4^*D_5^*D_6^*$. Hence, M is in $\mathsf{NCM}(\mathsf{SBD}_{id})$. Now define a homomorphism on $L(M)$ which maps a_0, a_1, a_2, a_3 to a and b_0, b_1, b_2 to b, c_0, c_1, c_2 to c, and d_0, d_1, d_2 to d. Then $L = h(L(M))$ is also in $\mathcal{L}(\mathsf{NCM}(\mathsf{SBD}_{id}))$. $\square$

Thus, SBD_{id} is enough to generate all bounded Ginsburg semilinear languages, whereas LB_{id} is not.

Next, we will explore the language families $\mathsf{NCM}(\mathsf{LB}_d)$ and $\mathsf{NCM}(\mathsf{LB}_i)$.

Proposition 17. *$\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_d))$ is the smallest full trio containing, for each $k \geq 1$, (here, $w_0 \in \Delta^*_{(k,c)}$, and $w_k \in D_k^*$),*

$$L_k^{\mathsf{LB}_d} = \{w_0 w_1 \cdots w_k \mid \begin{array}{l} w_i \in \{C_{i+1}, C_{i+2}, \ldots, C_k, D_i\}^*, 0 \leq i \leq k, \\ |w_0 w_1 \cdots w_{j-1}|_{C_j} = |w_j|_{D_j} > 0, 1 \leq j \leq k\} \subseteq \Delta_k^*. \end{array}$$

Proof. First, $L_k^{\mathsf{LB}_d}$ can be accepted by $M \in \mathsf{NCM}(\mathsf{LB}_d)$ by guessing a partition into $w_0 w_1 \cdots w_k$, and while reading w_i, verify it is in $\{C_{i+1}, C_{i+2}, \ldots, C_k, D_i\}^*$, incrementing counter j for every C_j read, and decrementing counter i for every D_i read, finishing with all counters empty.

From Lemma 1 and Proposition 2, it follows that $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_d))$ is the smallest full trio containing all $I \subseteq \Delta_k^*, k \geq 1$ such that

$$I = \{w \mid \begin{array}{l} w \in Y \sqcup Z, Y = \Delta^*_{(k,c)}, Z = D^*_{i_1} \cdots D^*_{i_k}, D_{i_1}, \ldots, D_{i_k} \text{ is a permutation} \\ \text{of } \Delta_{(k,d)}, |w|_{C_i} = |w|_{D_i} > 0, \text{ all } C_i\text{'s appear before any } D_i\text{'s}, \\ \text{for } 1 \leq i \leq k\}. \end{array}$$

Notice, every such I can be obtained from

$$I' = \{w \mid \begin{array}{l} w \in Y \sqcup Z, Y = \Delta^*_{(k,c)}, Z = D_1^* \cdots D_k^*, |w|_{C_i} = |w|_{D_i} > 0, \text{ all} \\ C_i\text{'s appear before any } D_i\text{'s, for } 1 \leq i \leq k\}. \end{array}$$

via homomorphisms that permute elements of $\Delta_{(k,d)}$ such that $h(D_i) = D_j$ implies $h(C_i) = C_j$.

Notice that there is only one such I' for each k, and it is equal to $L_k^{\mathsf{LB}_d}$. $\square$

The next proposition follows with a similar proof.

Proposition 18. *$\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_i))$ is the smallest full trio containing, for each $k \geq 1$, (here, $w_0 \in C_1^*$, and $w_k \in \Delta^*_{(k,d)}$),*

$$L_k^{\mathsf{LB}_i} = \{w_0 \cdots w_k \mid \begin{array}{l} w_i \in \{D_1, \ldots, D_i, C_{i+1}\}^*, 0 \leq i \leq k, \\ |w_{j-1}|_{C_j} = |w_j w_{j+1} \cdots w_k|_{D_j} > 0, 1 \leq j \leq k\} \subseteq \Delta_k^*. \end{array}$$

5 Applications To Existing Families

We will apply the results of this paper to quickly characterize the bounded languages inside known language families. It has been recently shown that finite-index $\mathsf{ET0L}$ languages contain all bounded Ginsburg semilinear languages [7]. This implies $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_{id})) \subseteq \mathcal{L}(\mathsf{ET0L}_{\text{fin}})$ (the family of finite-index $\mathsf{ET0L}$ languages, which is a full trio [12]; we refer to this paper for the formal definitions of $\mathsf{ET0L}$ systems and languages), and the bounded languages within $\mathsf{DCM}, \mathsf{NCM}$, and $\mathsf{ET0L}_{\text{fin}}$ coincide by Proposition 15. Here, we strengthen this result. First, it is shown that each $L_k^{\mathsf{LB}_d}$ and $L_k^{\mathsf{LB}_i}$ is in $\mathcal{L}(\mathsf{ET0L}_{\text{fin}})$.

Lemma 4. *For each $k \geq 1$, $L_k^{\mathsf{LB}_d}, L_k^{\mathsf{LB}_i} \in \mathcal{L}(\mathsf{ET0L}_{\text{fin}})$.*

Proof. Let $k \geq 1$. We can create an $\mathsf{ET0L}$ system of index $k+1$ to accept $L_k^{\mathsf{BD}_d}$ as follows. Let $G = (V, \mathcal{P}, S, \Delta_k)$. G has production tables and productions defined as follows (in the productions, C_j and D_j are terminals, and it is assumed that $C_j \to C_j$ and $D_j \to D_j$ are in every table):

- P_S contains $S \to S_0 S_1 \cdots S_k$.
- P^i_j for all $0 \le i < j \le k$, contains
  - $S_l \to S_l$, for all $l \in \{1, \dots, k\} - \{i, j\}$
  - $S_j \to D_j S_j$,
  - $S_i \to C_j S_i$.
- P_F contains $S_l \to \lambda$, for all $l, 0 \le l \le k$.

We will prove that $L(G) = L_k^{\mathsf{LB}_d}$.

"$\subseteq$" Let $w \in L(G)$. Thus, there exists a sequence of production tables $P_S, P^{i_1}_{j_1}, \dots, P^{i_n}_{j_n}, P_F$, such that $S \Rightarrow_{P_S} x_0 \Rightarrow_{P^{i_1}_{j_1}} x_1 \cdots \Rightarrow_{P^{i_n}_{j_n}} x_n \Rightarrow_{P_F} w$, and $x_0 = S_0 \cdots S_k$. It is clear that $x_0, \dots, x_n$ all have $S_0, \dots, S_k$ as the sequence of nonterminals. Let w_j be the sequence of terminals derived from S_j in w. Then only w_j can contain D_j by the productions, and w_j can only contain $C_{j+1}, \dots, C_k$ and not $C_1, \dots, C_j$ by the productions. Furthermore, for every D_j derived using table P^i_j say (implying $i < j$), only one other nonterminal, S_i, can derive a new symbol from $\Delta_{(k,c)}$, and it must be C_j which must get added to $w_i, i < j$. Hence, $w = w_0 w_1 \cdots w_k \in {}_k^{\mathsf{LB}_d}$.

"$\supseteq$" Let $w \in L_k^{\mathsf{LB}_d}$. Then $w = w_0 \cdots w_k, w_i \in \{C_{i+1}, \dots, C_k, D_i\}^*, 0 \le i \le k, |w_0 \cdots w_{j-1}|_{C_j} = |w_j|_{D_j} > 0, 1 \le j \le k$. We will show by induction that there is a sequence of word sequences (setting $n = |w|/2$) for $0 \le i \le n$, $\alpha_i = (w_{i,0}, \dots, w_{i,k})$, where $w_{i,j}$ is a prefix of w_j, $w_{0,j} = \lambda$, for $1 \le j \le k$, α contains the same number of C_j's as D_j's, for $1 \le j \le k$, and α_{i+1} differs from α_i in exactly two positions, $0 \le l < j \le k$, $w_{i+1,j} = w_{i,j} D_j, w_{i+1,l} = w_{i,l} C_j$.

The base case, $i = 0$, is true since $w_{0,j} = \lambda$ is a prefix of w_j trivially for each j, $0 \le j \le k$. Assume there exists $\alpha_i, i < n$, where $w_{i,j}$ is a prefix of w_j for all $0 \le j \le k$, and α_i contains the same number of C_j's as D_j's, for $1 \le j \le k$. Since $i < n$, there must exist some j' maximal, where $w_{i,j'}$ is not equal to $w_{j'}$. Then the "next letter" of $w_{j'}$ (ie. $(w_{i,j'})^{-1} w_{j'}$) must be $D_{j'}$, otherwise it is C_x, for some $x > j'$, but then one copy of D_x would yet to have been generated by the inductive hypothesis as D_x would need to occur in w_x, but $w_{i,x} = w_x$, by the maximality of j', a contradiction. Let l' be the largest number less than j' such that the next letter of $w_{l'}$, $(w_{i,l'})^{-1} w_{l'}$ is in $\Delta_{(k,c)}$, say $C_y, y > l'$. We will argue this must exist, otherwise all of w_0 would have been already consumed, as w_0 only contains symbols from $\Delta_{(k,c)}$, therefore all D_1's would have been consumed by the inductive hypothesis. But then w_1 must have been consumed, since the next letter does not start with a letter from $\Delta_{(k,c)}$ and it does not contain D_1. Then, the first letter remaining in any of $w_0, \dots, w_{l'-1}$ must be from $\Delta_{(k,d)}$, and therefore does not match a symbol from $\Delta_{(k,c)}$, a contradiction. Then, the remaining part of $w_{l'}$ must start with C_z for some $z > l'$, and then w_z must start with D_z since it does not start with any symbol of $\Delta_{(k,c)}$. Hence, the induction follows. Hence, $S \Rightarrow_{P_S} w_{0,0} S_0 \cdots w_{0,k} S_k \Rightarrow_{P^{l_1}_{j_1}} \cdots \Rightarrow_{P^{l_n}_{j_n}} w_{n,0} S_0 \cdots w_{n,k} S_k \Rightarrow_{P_F} w_{n,0} w_{n,1} \cdots w_{n,k} = w_0 w_1 \cdots w_k = w$, where α_{i+1} differs from α in positions l_i and j_i, $l_i < j_k$, for all i, $0 \le i < n$.

For $L_k^{\mathsf{LB}_i}$, this follows since it can be obtained by reversal and homomorphism from $L_k^{\mathsf{LB}_d}$, and finite-index $\mathsf{ET0L}$ is closed under these operations [12]. $\square$

It was also shown that that there are $\mathcal{L}(\mathsf{ET0L}_{\text{fin}})$ languages that are not in $\mathcal{L}(\mathsf{NCM})$ [7]. Then, this sub-family of $\mathcal{L}(\mathsf{NCM})$ is strictly contained in $\mathcal{L}(\mathsf{ET0L}_{\text{fin}})$.

Proposition 19. $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_\cup)) \subsetneq \mathcal{L}(\mathsf{ET0L}_{\text{fin}})$, *and* $\mathcal{L}(\mathsf{ET0L}_{\text{fin}})^{\text{bd}} = \mathcal{L}(\mathsf{DCM})^{\text{bd}}$.

Proof. Inclusion follows directly from Proposition 17 and Lemma 4, and since $\mathcal{L}(\mathsf{ET0L}_{\text{fin}})$ is a semilinear full trio [12]. Strictness follows from [7]. The fact that bounded languages are the same follows from Proposition 15. $\square$

We leave as an open problem whether there are languages accepted by NCM that cannot be generated by a finite-index $\mathsf{ET0L}$ system. We conjecture that over $\Sigma_k = \{a_1, \dots, a_k\}$, $\{w \mid |w|_{a_1} = \cdots = |w|_{a_k}\}$ is not in $\mathcal{L}(\mathsf{ET0L}_{\text{fin}})$, for some k. One might think that the (non-finite-index $\mathsf{ET0L}$) one-sided Dyck language on one letter is a candidate witness, but this language is not in $\mathcal{L}(\mathsf{NCM})$ [4].

Next, the class of TCA machines are Turing machines with a one-way read-only input tape, and a finite-crossing[1] read/write worktape. This language family is a semilinear full trio [5]. Therefore, $\mathcal{L}(\mathsf{TCA})^{\text{bd}} \subseteq \mathcal{L}(\mathsf{NCM})^{\text{bd}}$. To show that there is equality, we will simulate $\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d)$. Let M be a well-formed k-counter machine satisfying instruction language $I \subseteq w_1^* w_2^* \cdots w_l^* D_{i_1}^* \cdots D_{i_n}^*, w_i \in \Delta_{(k,c)}^*, 1 \leq i \leq l, D_j \in \Delta_{(k,d)}, 1 \leq j \leq n$. Then we build a TCA machine M' with worktape alphabet Δ_k that, on input w, simulates a derivation of M, whereby, if M increases from counters in the sequence $C_{j_1}, \dots, C_{j_m}$, M' instead writes this sequence on the worktape. Then, M' simulates the decreasing transitions of M as follows: for every section of decreases in $D_{i_j}^*$, for $1 \leq j \leq n$, M' sweeps the worktape from right-to-left, and corresponding to every decrease, replaces the next C_{i_j} symbol with the symbol D_{i_j} (thereby marking the symbol). This requires n sweeps of the worktape, and M' accepts if all symbols end up marked and the simulated computation is in a final state.

Proposition 20. $\mathcal{L}(\mathsf{NCM}(\mathsf{BD}_i\mathsf{LB}_d)) \subseteq \mathcal{L}(\mathsf{TCA})$*, and* $\mathcal{L}(\mathsf{TCA})^{\text{bd}} = \mathcal{L}(\mathsf{DCM})^{\text{bd}}$.

Next, the family of multi-push-down automata and languages has been introduced [1]. We let MP be these machines. They have some number k of pushdowns, and allow to push to every pushdown, but only pop from the first non-empty pushdown. This can clearly simulate every machine in $\mathsf{NCM}(\underline{\mathsf{LB}}_d)$ (distinct-letter-bounded, which is enough to accept every language in $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_d))$ by Lemma 1). Furthermore, it follows from results within [1] that $\mathcal{L}(\mathsf{MP})$ is closed under reversal (since it is closed under homomorphic replication with reversal, and homomorphism). Therefore, $\mathcal{L}(\mathsf{MP})$ also contains $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_\cup))$. Also, this family only contains semilinear languages [1]. Therefore, the bounded languages within $\mathcal{L}(\mathsf{MP})$ coincide with those in $\mathcal{L}(\mathsf{NCM})$ and $\mathcal{L}(\mathsf{DCM})$.

Proposition 21. $\mathcal{L}(\mathsf{NCM}(\mathsf{LB}_\cup)) \subseteq \mathcal{L}(\mathsf{MP})$*,* $\mathcal{L}(\mathsf{MP})^{\text{bd}} = \mathcal{L}(\mathsf{DCM})^{\text{bd}}$.

References

[1] L. Breveglieri, A. Cherubini, C. Citrini, and S.C. Reghizzi. Multi-push-down languages and grammars. *International Journal of Foundations of Computer Science*, 7(3):253–291, 1996.

[2] S Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages*. North-Holland Publishing Company, Amsterdam, 1975.

[3] Seymour Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, Inc., New York, NY, USA, 1966.

[4] S. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7:311–324, 1978.

[5] Tero Harju, Oscar Ibarra, Juhani Karhumäki, and Arto Salomaa. Some decision problems concerning semilinearity and commutation. *Journal of Computer and System Sciences*, 65(2):278–294, 2002.

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

[7] O.H. Ibarra and I. McQuillan. On bounded semilinear languages, counter machines, and finite-index ET0L, 2016. accepted to the 21st International Conference on Implementation and Application of Automata (CIAA).

[8] O.H. Ibarra and B. Ravikumar. On bounded languages and reversal-bounded automata. *Information and Computation*, 246(C):30–42, 2016.

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

[10] Oscar H. Ibarra and Shinnosuke Seki. Characterizations of bounded semilinear languages by one-way and two-way deterministic machines. *International Journal of Foundations of Computer Science*, 23(6):1291–1306, 2012.

[11] J. Kortelainen and T. Salmi. There does not exist a minimal full trio with respect to bounded context-free languages. In G. Mauri and A. Leporati, editors, *Lecture Notes in Computer Science*, volume 6795 of *15th International Conference on Developments in Language Theory, DLT 2011, Milan, Italy*, pages 312–323, 2011.

[12] G. Rozenberg and D. Vermeir. On ET0L systems of finite index. *Information and Control*, 38:103–133, 1978.