On the Containment Problem for Deterministic Multicounter Machine Models

Oscar H. Ibarra[1] and Ian McQuillan[2(✉)]

[1] Department of Computer Science, University of California, Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu
[2] Department of Computer Science, University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada
mcquillan@cs.usask.ca

Abstract. A new model of one-way multicounter machines is introduced. In this model, within each transition, testing the counter status of a counter is optional, rather than existing models where they are always either required (traditional multicounter machines) or no status can be checked (partially-blind multicounter machines). If, in every accepting computation, each counter has a bounded number of sections that decrease that counter where its status is tested, then the machine is called *finite-testable*. One-way nondeterministic finite-testable multicounter machines are shown to be equivalent to partially-blind multicounter machines, which, in turn, are known to be equivalent to Petri net languages and languages defined by vector addition systems with states. However, one-way deterministic finite-testable multicounter machines are strictly more general than deterministic partially-blind machines. Moreover, they also properly include deterministic reversal-bounded multicounter machines (unlike deterministic partially-blind multicounter machines). Interestingly, one-way deterministic finite-testable multicounter machines are shown to have a decidable containment problem ("given two machines M_1, M_2, is $L(M_1) \subseteq L(M_2)$?"). This makes it the most general known model where this problem is decidable. We also study properties of their reachability sets.

1 Introduction

One of the most commonly studied decision problems for models of automata is the containment problem (also sometimes called the inclusion problem), which is: "given two machines M_1 and M_2 from the model, is $L(M_1) \subseteq L(M_2)$?". The containment problem is important towards model checking. Indeed, if M_2 contains an automaton-based representation of a specification and M_1 contains a model, then M_1 satisfies the specification if $L(M_1) \subseteq L(M_2)$. This automata-theoretic

The research of I. McQuillan was supported, in part, by Natural Sciences and Engineering Research Council of Canada.

É. André and J. Sun (Eds.): ATVA 2023, LNCS 14215, pp. 74–94, 2023.
https://doi.org/10.1007/978-3-031-45329-8_4

approach was initiated by Vardi and Wolper [28], it enables on-the-fly model checking [8], and it has been studied with different models of automata [23,27]. Furthermore, industrial automated verification tools have implemented and used automata-based methods [11]. Not only is the containment problem undecidable for the context-free languages, it is also undecidable for many restriction of pushdown automata, including deterministic pushdown automata, deterministic one counter automata (the store contains a non-negative integer which can be increased, decreased, and tested for zero), and nondeterministic one counter automata where the counter cannot increase after decreasing [1]. In contrast, one model with a decidable containment problem is one-way deterministic machines with some number of counters, but on every accepting computation, there is a bound in the number of changes in direction between non-decreasing and non-increasing the size of each counter, called *reversal-bounded*, and they are denoted by DRBCM (and the nondeterministic version is denoted by NRBCM).

The emptiness problem, "given machine M, is $L(M) = \emptyset$?", is also important and commonly decidable for more powerful models. Indeed, it is decidable for pushdown automata [12], NRBCM [17], and one-way nondeterministic partially-blind multicounter machines (denoted by NPBLIND) [10]. The latter model contains multicounter machines where each counter contains some non-negative integer, but no differences are allowed in available transitions based on the counter status (whether a counter is empty or not), besides acceptance being defined by final state and all counters being zero in the final configuration. In this sense, counter status checks are not allowed. It is known that NRBCM is properly contained in NPBLIND [10], and also that the following are equivalent: deciding emptiness for partially-blind multicounter machines, deciding the emptiness problem for Petri nets, and deciding reachability of vector addition systems. Later, reachability for Petri nets was shown to be decidable and therefore all three problems are decidable [20,22]. Recently, it was shown that the boundedness problem ("given M, are there words $w_1, \ldots, w_n$ such that $L(M) \subseteq w_1^* \cdots w_n^*$?") is decidable for vector addition systems with states [6], hence for NPBLIND as well.

Some restrictions of NPBLIND (resp. labelled Petri nets, and vector addition systems with states) have also been studied. For example, λ-free deterministic labelled Petri nets have been studied [25,29]. In the latter paper, it was shown that the complement of the language accepted by any λ-free deterministic labelled Petri net could be accepted by a nondeterministic labelled Petri net (equivalent to NPBLIND). From this, and decidability of emptiness for Petri nets, it follows that the containment problem is decidable for λ-free deterministic labelled Petri nets. To note here, this type of Petri net also does not have an explicit label to detect when it has reached the end of the input, which can limit the capacity of the machines. In addition, vector addition systems with states that are boundedly-ambiguous have been studied [5]. When using an acceptance condition defined by an upward-closed set of configurations, the containment problem is decidable.

Here, we study deterministic NPBLIND machines with the input end-marker and also allowing λ transitions, which we denote by DPBLIND. We show that

the right input-marker strictly increases the capacity, showing the importance of using this simple construct. Even with λ transitions and the end-marker however, we show the model is still somewhat limited and cannot accept all DRBCM languages.

This inspires a simple and novel restriction of multicounter machines, where the counter status checks are optional. Such a machine is r-testable if, in every accepting computation, each counter has at most r segments where it decreases this counter and checks its status at least once (and it is finite-testable if it is r-testable for some r). This class is denoted by NTCM, and DTCM for the deterministic restriction. While we show that NTCM and NPBLIND are equivalent, DTCM is shown to be strictly more powerful than both DPBLIND and DRBCM. Then we show that the complement of every DTCM (hence DPBLIND) is in NTCM = NPBLIND. From this, it follows that DTCM (and DPBLIND) has a decidable containment problem (hence also equality problem and universe problem). This makes DTCM one of the most general automata model known with a decidable containment problem, as it is significantly more general than DRBCM and allows λ transitions and has an end-marker unlike the model [25].

All omitted proofs appear in the appendix due to space constraints.

2 Preliminaries

Let $\mathbb{N}$ be the set of natural numbers, and $\mathbb{N}_0$ be the set of non-negative integers, and $\mathbb{Z}$ be the set of integers. For $k \in \mathbb{N}$, let $\mathbb{N}(k) = \{1, \ldots, k\}$. For $j \in \mathbb{N}_0$, define $\pi(j)$ to be $\hat{0}$ if $j = 0$ and $\hat{1}$ otherwise. For a set X and $k \in \mathbb{N}$, define X^k to be the set of k-tuples of elements of X. A set $Q \subseteq \mathbb{N}_0^k$ is called a *linear set* if there exists vectors $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_l \in \mathbb{N}_0^k$ such that $Q = \{\vec{v}_0 + i_1\vec{v}_1 + \cdots + i_l\vec{v}_l \mid i_1, \ldots, i_l \in \mathbb{N}_0\}$. Here, $\vec{v}_0$ is called the *constant*, and $\vec{v}_1, \ldots, \vec{v}_l$ are called the *periods*. The constant and the periods together are called a *representation* of the linear set. A set $Q \subseteq \mathbb{N}_0^k$ is called a *semilinear set* if it is a finite union of linear sets, and a representation of Q is the set of representations of each linear set.

We assume a basic familiarity with automata and formal language theory [12]. Let Σ be a finite alphabet, and let Σ^* be the set of all words over Σ, including the empty word λ, and Σ^+ is the set of all non-empty words. A *language over* Σ is any $L \subseteq \Sigma^*$. Given $L \subseteq \Sigma^*$, the *complement* of L with respect to Σ is, $\overline{L} = \Sigma^* - L$. Given a word $w \in \Sigma^*$, $|w|$ is the length of w; and given $a \in \Sigma$, $|w|_a$ is the number of a's in w.

Given a fixed ordering of an alphabet $\Sigma = \{a_1, \ldots, a_k\}$, then the *Parikh image* of $w \in \Sigma^*$ denoted $\Psi(w) = (|w|_{a_1}, \ldots, |w|_{a_k})$. This is extended to the Parikh image of languages $L \subseteq \Sigma^*$ by $\Psi(L) = \{\Psi(w) \mid w \in L\}$. A language L is said to be *Parikh semilinear* (or simply *semilinear*) if $\Psi(L)$ is a semilinear set. It is known that every regular language (in fact, context-free language) is Parikh semilinear [24]. We say that a set for a given problem is an *effectively determinable semilinear set* if, the set for that problem is a semilinear set, and moreover there is an effective procedure to determine the semilinear representation given inputs to the problem.

A language $L \subseteq \Sigma^*$ is *bounded* if there exists $w_1, \ldots, w_n \in \Sigma^+$ such that $L \subseteq w_1^* \cdots w_n^*$. Given words $w_1, \ldots, w_n \in \Sigma^+$ and $L \subseteq w_1^* \cdots w_n^*$, we define a function ϕ from languages L to subsets of $\mathbb{N}_0^n$ that maps $\phi(L) = \{(i_1, \ldots, i_n) \mid w_1^{i_1} \cdots w_n^{i_n} \in L\}$. We call L a *bounded Ginsburg semilinear* language (or simply Ginsburg semilinear) if $\phi(L)$ is a semilinear set. In the literature, bounded Ginsburg semilinear is often just referred to by bounded semilinear [3]. Note that the Parikh image $\psi(L)$ (which is a subset of $\mathbb{N}_0^k$) may be different than $\phi(L)$ (which is a subset of $\mathbb{N}_0^n$). However, it is known that every bounded Ginsburg semilinear language is also Parikh semilinear [3] but there are Parikh semilinear languages (even bounded ones) that are not Ginsburg semilinear. For example, consider $L = \{a^{2^n} b \mid n > 0\} \cup ba^+$, which is bounded as it is a subset of $a^*b^*a^*$. But given a, b, a, $\phi(L) = \{(2^n, 1, 0) \mid n > 0\} \cup \{(0, 1, n) \mid n > 0\}$, which is not semilinear, and so L is not Ginsburg semilinear. However, $\psi(L) = \{(n, 1) \mid n > 0\}$ (a is first letter, b is second) is semilinear and so L is Parikh semilinear.

We define k-counter machines in a slightly unusual way, where it is possible to either test whether a counter is positive or zero, but also not test the status of a counter. This allows the definition to be used for multiple purposes.

Definition 1. *A one-way nondeterministic k-counter machine is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ with a finite set of states Q, initial state q_0, the final state set $F \subseteq Q$, an input alphabet Σ, and a transition function δ, which is a partial function from $Q \times (\Sigma \cup \{\lambda, \lhd\}) \times \mathbb{N}(k) \times \{\hat{0}, \hat{1}, \hat{\varnothing}\}$ to finite subsets of $Q \times \{0, +1, -1\}$, where $\lhd \notin \Sigma$ is the right input end-marker. Here, we call $\{\hat{0}, \hat{1}, \hat{\varnothing}\}$ the set of* tests*, with $\hat{0}$ the* zero-test*, $\hat{1}$ the* non-zero-test*, $\hat{0}$ and $\hat{1}$ collectively the* status tests*, and $\hat{\varnothing}$ is the* no-test*. A transition $(p, e) \in \delta(q, a, i, \tau)$ (which we often write as $\delta(q, a, i, \tau) \to (p, e)$) can be used if q is the current state and $a \in \Sigma \cup \{\lambda, \lhd\}$ is read from the input, and*

- *if $\tau = \hat{\varnothing}$, then it adds e to counter i,*
- *if $\tau = \hat{1}$ and counter i is non-empty, then it adds e to counter i,*
- *if $\tau = \hat{0}$ and counter i is empty, then it adds e to counter i,*

and it switches to state p. Such a machine is deterministic *if,*

1. *for each $q \in Q$ and $a \in \Sigma \cup \{\lhd\}$, there is at most one counter i, denoted $C(q, a)$ if one exists, such that $\delta(q, a, i, \tau) \cup \delta(q, \lambda, i, \tau) \neq \emptyset$, for some $\tau \in \{\hat{0}, \hat{1}, \hat{\varnothing}\}$,*
2. *for all $q \in Q, a \in \Sigma \cup \{\lhd\}, \tau \in \{\hat{0}, \hat{1}\}$, where $i = C(q, a)$,*

$$|\delta(q, a, i, \tau) \cup \delta(q, \lambda, i, \tau) \cup \delta(q, a, i, \hat{\varnothing}) \cup \delta(q, \lambda, i, \hat{\varnothing})| \leq 1.$$

This matches the traditional notion of determinism, except the counter status can influence the deterministic choice of the next transition to apply if and only if a status test is used. If a no-test is used, it must be the only available transition. For example, a deterministic machine could have separate transitions from $\delta(q, a, i, \hat{0})$ and $\delta(q, a, i, \hat{1})$, where the first happens if counter i is zero and the second happens if counter i is positive. But we cannot have separate transitions if either both transitions have the same test, or if one is a no-test as that would lead to multiple possible transitions that could be applied from

the same instantaneous description (defined next). It is also required that only one counter can be used from a given state and input letter or empty word, as otherwise multiple instantaneous descriptions could follow a given one.

Definition 2. *An* instantaneous description (ID) *of k-counter machine $M = (Q, \Sigma, \delta, q_0, F)$ is a member of $Q \times (\Sigma^* \triangleleft \cup \{\lambda\}) \times \mathbb{N}_0^k$. Instantaneous descriptions change via the relation $\vdash_M$ (or $\vdash$ if M is clear) with $(q, aw, y_1, \ldots, y_k) \vdash (q', w, y_1, \ldots, y_{i-1}, y_i + e, y_{i+1}, \ldots, y_k)$, if $\delta(q, a, i, \tau) \to (q', e)$, $y_i + e \geq 0$, and either $\tau = \hat{\varnothing}$ or $\pi(y_i) = \tau$. Then, $\vdash^*$ is the reflexive, transitive closure of $\vdash$. A* computation *on $w \in \Sigma^*$ is a sequence of IDs,*

$$(p_0, w_0, y_{0,1}, \ldots, y_{0,k}) \vdash \cdots \vdash (p_n, w_n, y_{n,1}, \ldots, y_{n,k}), \tag{1}$$

where $q_0 = p_0, w_0 = w\triangleleft, y_{0,i} = 0, 1 \leq i \leq k$; and a computation is an accepting computation of w *if $y_{n,i} = 0$ for $1 \leq i \leq k, w_n = \lambda$, and $p_n \in F$. Thus, accepting computations end at all 0's in the counters and in a final state. Often, we associate labels bijectively from a set Σ_δ to the transitions of M, and write $ID \vdash^t ID'$, $t \in \Sigma_\delta$ if $ID \vdash ID'$ via transition t; and $ID \vdash^x ID'$ for $x = t_1 \cdots t_m, t_i \in \Sigma_\delta$, if $ID = ID_0 \vdash^{t_1} \cdots \vdash^{t_m} ID_m = ID'$. We also define* $\text{runs}(M) = \{x \in \Sigma_\delta^* \mid (q_0, w\triangleleft, 0, \ldots, 0) \vdash^x (q_f, \lambda, 0, \ldots, 0), q_f \in F\}$

Given a computation $ID_0 \vdash^{t_1} \cdots \vdash^{t_n} ID_n, n \geq 0$, for i $1 \leq i \leq k$, we divide it into so-called decreasing i-segments and increasing i-segments as follows: we say $ID_{j-1} \vdash^{t_j} \cdots \vdash^{t_l} ID_l$ is a decreasing i-segment *if*

- *t_j decreases counter i,*
- *there are no transitions that increase counter i in $t_j, \ldots, t_{l-1}$,*
- *either $l = n$ or t_l increases counter i,*
- *the last transition of $t_1, \ldots, t_{j-1}$ that changes counter i increases it.*

We can naturally order decreasing i-segments. Further, we define the increasing i-segments *between the beginning of the computation to the ID at the start of the first decreasing i-segment or the last ID if there are no decreasing i-segments, between the last ID of one decreasing i-segment and the first ID of the next decreasing i-segment, and from the last ID of the last decreasing i-segment to the end if it does not end in a decreasing i-segment or it ends with a transition that increases counter i.*

Such a machine M is r-reversal-bounded, *if, in every accepting computation, each counter i, $1 \leq i \leq k$, has at most $r+1$ increasing or decreasing i-segments. It is* reversal-bounded *if it is r-reversal-bounded for some r. Such a machine is called* partially-blind *if all transitions have $\hat{\varnothing}$ for tests.*

The language accepted by M,

$$L(M) = \{w \in \Sigma^* \mid \text{there is an accepting computation of } w\},$$

and the reachability set of M, $R(M) = \{(q, v_1, \ldots, v_k) \mid (q_0, w\triangleleft, 0, \ldots, 0) \vdash^* (q, w', v_1, \ldots, v_k) \vdash^* (q_f, \lambda, 0, \ldots, 0), q_f \in F\}$.

As decreasing i-segments are maximal (they start with a decrease, the previous transition that changes that counter is an increase, and they end with either the next increase or the end of the computation), we can split each computation, for each i, uniquely into increasing and decreasing i-segments. The definition of r-reversal-bounded here is equivalent to that of [17] which counts the number of changes in direction in each counter. The definition of partially-blind multicounter machines is the same to that of [10]. Note these machines do have one implicit test of all zeros in the counters at the end of the computation. With no-tests (and partially-blind machines), the machines "crash" (ie. the computation cannot continue), if any counter tries to go below zero. But the machines cannot detect that the counters are zero (because the transition function does not allow differences based on the contents of the counters). Normally the reachability set is defined without the restriction of appearing within an accepting computation, but we will use this stronger notion here. Also, sometimes it is defined to not include the state as a component. We will usually associate the state component bijectively with a number in $\{1, \ldots, |Q|\}$ so we can, e.g. talk about a reachability set being a semilinear set.

The class of one-way nondeterministic (resp. deterministic) partially-blind k-counter machines is denoted by $\mathsf{NPBLIND}(k)$ (resp. $\mathsf{DPBLIND}(k)$), and the class of partially-blind machines is denoted by $\mathsf{NPBLIND}$ (resp. $\mathsf{DPBLIND}$). The class of one-way nondeterministic (resp. deterministic) r-reversal-bounded k-counter machines is denoted by $\mathsf{NRBCM}(k, r)$ (resp. $\mathsf{DRBCM}(k, r)$), and the family of reversal-bounded multicounter machines is denoted by NRBCM (resp. DRBCM). By a slight abuse of notation, we will use the same notation for a class of machines and the family of languages they accept.

It is also known that one-way deterministic two-counter machines accept all recursively enumerable languages (denoted RE), but there are some recursively enumerable languages that are not in $\mathsf{NPBLIND}$ [10]. Deterministic partially-blind machines are a restriction of partially-blind machines, and it is therefore clear that $\mathsf{DPBLIND} \subseteq \mathsf{NPBLIND} \subsetneq \mathsf{RE}$. Lastly, note $\mathsf{DRBCM} \subsetneq \mathsf{NRBCM} \subsetneq \mathsf{NPBLIND} \subsetneq \mathsf{RE}$, with the latter two shown in [10], and DRBCM is known to be a proper subset of NRBCM [17].

Lastly, we show a simple result in this section which will help throughout the paper. Given a k-counter machine $M = (Q, \Sigma, \delta, q_0, F)$, denote by $M_\delta = (Q_\delta, \Sigma_\delta, \delta', q_0, F')$ the deterministic k-counter machine obtained from M with $Q_\delta = Q \cup Q'$ where Q' is a primed version of the states in Q, F' is the primed versions of the states in F, and δ' is built to read $t \in \Sigma_\delta$ to simulate transition t of M, but it uses states of Q to simulate transitions of M that read letters of $\Sigma \cup \{\lambda\}$, but M' instead reads t, and switches from states in Q to states of Q' if t reads $\lhd$ in M, and switches between states in Q' to simulate only λ transitions. Lastly, add $\delta'(q', \lhd, 1, \hat{\varnothing}) \to (q', 0)$ for all q', where $q \in Q$.

Lemma 3. *Given k-counter $M = (Q, \Sigma, \delta, q_0, F)$, the following are true:*

- *M_δ is deterministic,*
- $\mathrm{runs}(M) = L(M_\delta)$,
- *$L(M)$ is not Parikh semilinear implies* $\mathrm{runs}(M)$ *is not Parikh semilinear,*
- *$R(M)$ is not semilinear implies $R(M_\delta)$ is not semilinear.*

3 Properties of Deterministic Partially-Blind Machines

We start this section with an example of a DPBLIND machine before analyzing its properties.

Example 4. It is known that NPBLIND contains the so-called one-sided Dyck language on one letter, $D_2 = \{w \in \{a,b\}^* \mid |w|_a = |w|_b, \text{ and if } w = xy, \text{ then } |x|_a \geq |x|_b\}$, which cannot be accepted by any NRBCM [10].

A NPBLIND(1) machine $M = (Q, \Sigma, \delta, q_0, \{q_1\})$ that accepts this language is as follows,

$$\delta(q_0, a, 1, \hat{\varnothing}) \to (q_0, +1),\ \delta(q_0, b, 1, \hat{\varnothing}) \to (q_0, -1),\ \delta(q_0, \lhd, 1, \hat{\varnothing}) \to (q_1, 0).$$

This machine by definition is deterministic, and therefore $D_2 \in$ DPBLIND(1). Since Dyck languages are Parikh semilinear, this shows that DPBLIND contains Parikh semilinear languages that are not in NRBCM. Furthermore, consider the prefix closure of D_2, $D_2' = \{w \in \{a,b\}^* \mid \text{if } w = xy, \text{ then } |x|_a \geq |x|_b\}$. A DPBLIND M' can accept D_2' by adding transition $\delta(q_1, \lambda, 1, \hat{\varnothing}) \to (q_1, -1)$ to M above. Notice that when it reads the end-marker with some value j on the counter, it continually decreases the counter, and as long as it eventually passes over 0 on the counter in state q_1, it accepts.

The right input end-marker is not necessary for any nondeterministic machine model defined in the previous section, because the machine can guess that it has read the last input symbol, and only accept if it guessed correctly. One could define $\mathsf{NRBCM_{NE}}$ (resp. $\mathsf{DRBCM_{NE}}$) machines as being NRBCM machines (resp. DRBCM) without containing any transitions on the end-marker, and acceptance is defined as, it reads the entire input w (with no end-marker), and it is in a final state. This notation was used in [7,14] where machines without the end-marker were studied with NRBCM and DRBCM. Hence, $\mathsf{NRBCM} = \mathsf{NRBCM_{NE}}$. For that reason, when studying nondeterministic machines, we can leave off the end-marker in the machine definition and when examining computations. With deterministic machines though, it is not necessarily so; e.g. it is known that $\mathsf{DRBCM_{NE}} \subsetneq \mathsf{DRBCM}$ [7,14] (in contrast to say deterministic pushdown automata where they are the same). We do not know of any other one-way input machine model where they are known to be different. Clearly, for any deterministic model defined above, the family of languages accepted without the end-marker is a subset of the entire model as we could simply ignore the marker. So we by default use the more general definition with the end-marker. Deterministic partially-blind multicounter languages have been defined and studied previously, but they were defined without the end-marker [4], and so we are using the more general definition with the end-marker here instead. We use the notation $\mathsf{NPBLIND_{NE}}$ (resp. $\mathsf{DPBLIND_{NE}}$) to be NPBLIND (resp. DPBLIND) machines and languages without the end-marker, where acceptance occurs by hitting the end of the input in a final state with all counters zero. Then $\mathsf{NPBLIND_{NE}} = \mathsf{NPBLIND}$ using the same argument as the start of this section.

We next see that in fact the language D_2' from Example 4 requires the end-marker to accept for deterministic machines, leading to the following separation.

Proposition 5. $\mathsf{DPBLIND}_{\mathsf{NE}} \subsetneq \mathsf{DPBLIND}$.

Therefore, the right end-marker increases the power of DPBLIND, similarly to DRBCM.

Next, we analyze whether DPBLIND and $\mathsf{DPBLIND}_{\mathsf{NE}}$ contain languages that are not Parikh semilinear, and whether reachability sets can be non-semilinear.

Proposition 6. DPBLIND *and* $\mathsf{DPBLIND}_{\mathsf{NE}}$ *both contain machines* M *such that* $L(M)$ *is not Parikh semilinear. Furthermore, both contain machines* M *such that* $R(M)$ *and* $\mathrm{runs}(M)$ *are not semilinear.*

Proof. It is known that NPBLIND contains languages that are not Parikh semilinear (Theorem 4 of [10]). Let $M \in \mathsf{NPBLIND}$ accepting any such language. Considering M_δ, $M_\delta \in \mathsf{DPBLIND}_{\mathsf{NE}}$ by Lemma 3. Also Lemma 3 indicates that $L(M_\delta) = \mathrm{runs}(M)$ is not Parikh semilinear. For the second point, it is known that there are $M \in \mathsf{NPBLIND}$ with $R(M)$ not being semilinear (implied by being the case for vector addition systems with states [13]). Using M_δ, Lemma 3 implies $R(M_\delta)$ is not semilinear. □

Hence, DPBLIND, $\mathsf{DPBLIND}_{\mathsf{NE}}$, and more general models introduced later in this paper all have languages, 'runs', and reachability sets that are not semilinear. In contrast, languages accepted by NRBCM are all semilinear [17], the 'runs' are semilinear by Lemma 3, and the reachability sets are semilinear, seen as follows: for each $M \in \mathsf{NRBCM}(k)$, create $M' \in \mathsf{NRBCM}(2k)$ that nondeterministically guesses and simulates transitions of M but using λ input and by using two identical sets of counters until a nondeterministically guessed spot. Then, it verifies that the input is $1^n c_1^{i_1} \cdots c_k^{i_k}$ where n is a number associated with the current state, and $i_1, \ldots, i_k$ are the same as one copy of the counters. From then, it continues the simulation using the other set of counters. It is evident that $\psi(L(M')) = R(M)$, which is semilinear [17].

Even though DPBLIND contains languages that are not Parikh semilinear, we see next that DPBLIND is still somewhat limited and cannot accept some languages that seem relatively simple and that are Ginsburg semilinear and can even be accepted by a $\mathsf{DRBCM}(1,1)$. By contrast DRBCM accepts all Ginsburg semilinear languages [19].

Proposition 7. *The language* $L = \{a^l b^m \mid 0 < l < m\}$ *is in* $\mathsf{DRBCM}(1,1)$ *but not in* DPBLIND. *Thus,* DPBLIND *does not contain all Ginsburg semilinear languages.*

Corollary 8. *The families* DPBLIND *(resp.* $\mathsf{DPBLIND}_{\mathsf{NE}}$*) and* DRBCM *are incomparable.*

The one direction follows from Proposition 7, and the other since DPBLIND and $\mathsf{DPBLIND}_{\mathsf{NE}}$ contain languages that are not Parikh semilinear by Proposition 6, but DRBCM does not [17].

Proposition 9. DPBLIND *is not closed under complement.*

Proof. Assume otherwise. Consider L from Proposition 7, and let $L' = \{a^n b^m \mid n \geq m\}$. We can see that L' can be accepted by a DPBLIND(1) machine that adds to the counter for each a, then subtracts for each b, and then at the end-marker, switches to final state and continually decreases the counter. Also, DPBLIND is clearly closed under intersection with regular languages. But $\overline{L'} \cap a^+b^+ = L \in$ DPBLIND, but this is not in DPBLIND by Proposition 7, a contradiction. □

Despite not being closed under complement, we will see later that the complement of every DPBLIND language is in NPBLIND, which is sufficient to show that DPBLIND has a decidable containment problem; in fact, we will determine a stronger result.

4 Finite-Testable Counter Machines

Next, we introduce a new restriction of counter machines, which will be the focus of the rest of this paper. A k-counter machine $M = (Q, \Sigma, \delta, q_0, F)$ is *r-testable* if, for every acceptable computation and every counter i, $1 \leq i \leq k$, there are at most r decreasing i-segments that contain at least one status test. A machine is *finite-testable* if it is r-testable for some $r \geq 0$. We denote the class of one-way nondeterministic (resp. deterministic) r-testable k-counter machines by $\mathsf{NTCM}(k, r)$ (resp. $\mathsf{DTCM}(k, r)$). We use $\mathsf{NTCM}(*, r)$ (resp. $\mathsf{DTCM}(*, r)$) for r-testable k-counter machines for some k. We also use NTCM (resp. DTCM) to refer to all one-way nondeterministic (resp. deterministic) finite-testable multi-counter machines.

Note, we could have alternatively defined r-testable so that for every accepted word, there is some accepting computation where each counter i has at most r decreasing i-segments; had we done that, given any machine M, another machine M' could be constructed that uses the finite control to count the number of decreasing i-segments in each counter, thereby satisfying the definition we use.

It is immediate that every NRBCM (resp. DRBCM) is a NTCM (resp. DTCM), and therefore NRBCM $\subseteq$ NTCM and DRBCM $\subseteq$ DTCM.

Example 10. It is evident that DPBLIND $\subsetneq$ DTCM since $L = \{a^l b^m \mid 0 < l < m\}$ is in DRBCM(1, 1) but not DPBLIND by Proposition 7. A machine $M \in$ DTCM accepting L contains the following transitions (q_f is a final state):

$$\delta(q_0, a, 1, \hat{0}) \to (q_1, +1), \qquad \delta(q_2, b, 1, \hat{0}) \to (q_3, 0),$$
$$\delta(q_1, a, 1, \hat{1}) \to (q_1, +1), \qquad \delta(q_3, b, 1, \hat{0}) \to (q_3, 0)$$
$$\delta(q_1, b, 1, \hat{1}) \to (q_2, -1), \qquad \delta(q_3, \lhd, 1, \hat{0}) \to (q_f, 0),$$
$$\delta(q_2, b, 1, \hat{1}) \to (q_2, -1)$$

Example 11. Next we provide a more complicated example of a machine accepting a language that cannot be accepted by an NRBCM. Recall $D_2' = \{w \in \{a,b\}^* \mid \text{if } w = xy, \text{ then } |x|_a \geq |x|_b\}$ from Example 4 which is not in NRBCM. Let L_1 be D_2' over $\{a_1, b_1\}$ and L_2 be D_2' over $\{a_2, b_2\}$. Let

$$L = \{u_1 v_1 \cdots u_l v_l \$ x_1 y_1 \cdots x_n y_n \mid u_1 \cdots u_l x_1 \cdots x_n \in L_1, v_1 \cdots v_l y_1 \cdots y_n \in L_2, \text{ and } |u_1 \cdots u_l|_{a_1} = |u_1 \cdots u_l|_{b_1}\}.$$

A DTCM$(2,2)$ $M = (Q, \Sigma, \delta, q_0, \{q_f\})$ that accepts this language is as follows, for $i \in \{1,2\}$,

$$\begin{array}{ll} \delta(q_0, a_i, i, \hat{\varnothing}) \to (q_0, +1), & \delta(q_1, \lhd, 1, \hat{\varnothing}) \to (q_2, 0), \\ \delta(q_0, b_i, i, \hat{\varnothing}) \to (q_0, -1), & \delta(q_2, \lambda, 1, \hat{1}) \to (q_2, -1) \\ \delta(q_0, \$, 1, \hat{0}) \to (q_1, 0), & \delta(q_2, \lambda, 1, \hat{0}) \to (q_3, 0), \\ \delta(q_1, a_i, i, \hat{\varnothing}) \to (q_1, +1) & \delta(q_3, \lambda, 2, \hat{1}) \to (q_3, -1) \\ \delta(q_1, b_i, i, \hat{\varnothing}) \to (q_1, -1) & \delta(q_3, \lambda, 2, \hat{0}) \to (q_f, 0). \end{array}$$

This machine is deterministic because from both q_0 and q_1 and on each letter of a_1, b_1, a_2, b_2, only a single no-test transition is possible, and from q_2 and q_3, only one transition on each status test is possible.

It also appears that this language cannot be accepted by a DPBLIND because a DPBLIND cannot test for zero until the very last ID. While it is possible to create two new counters (called the special counters) to count both $|u_1 \cdots u_l|_{a_1}$ and $|u_1 \cdots u_l|_{b_1}$ by counting until \$ and then not changing those counters until the end-marker, it seems not possible to deterministically decrease these special counters to zero to test that they are equal in the final instantaneous description while also decreasing the other counters to zero without the ability to test for zero in a subset of the counters before the final instantaneous description. In contrast, the DTCM machine above can detect whether an individual counter is empty unlike NPBLIND, which M does after reading \$, and also separately for each counter after reading $\lhd$.

Notice that NTCM$(*,0)$ corresponds exactly to NPBLIND. Furthermore, we see that with nondeterministic machines, finite-testability and 0-testability are equivalent, although converting to 0-testable increases the number of counters.

Proposition 12. NTCM$(*,0)$ = NPBLIND = NTCM.

For the rest of this section, we are only concerned with deterministic machines. We start with a normal form that is useful for the next section. We say a DTCM(k,r) $M = (Q, \Sigma, \delta, q_0, F)$ is in *normal form* if, for each $q \in Q, a \in \Sigma \cup \{\lhd\}$, the following are equivalent:

- there is a transition in $\delta(q, b, i, \hat{0})$ with $b \in \{a, \lambda\}$,
- there is a transition in $\delta(q, b, i, \hat{1})$ with $b \in \{a, \lambda\}$,
- there is no transition in $\delta(q, b, i, \hat{\varnothing})$ with $b \in \{a, \lambda\}$.

Furthermore, M is in *strong normal form* if M is in normal form, and

- $Q = Q_1 \cup Q_2, Q_1 \cap Q_2 = \emptyset$, Q_2 only contains λ transitions, and in every computation, M only uses transition from Q_1 before reading $\lhd$, and from Q_2 after reading $\lhd$,
- in every computation, on counter i, at most r successful zero-tests on counter i are possible.

Remark 13. Here, normal form enforces that on each letter of $\Sigma \cup \{\lhd\}$, there is a transition that can be applied with a zero test if and only if there is a transition that can be applied with a non-zero test, and vice versa (which, by determinism would imply that there is not a no-test transition that can be applied). If neither is true, a no-test is available. With strong normal form, as there are only λ transitions from states in Q_2, the normal form rules enforces that for each $q \in Q_2$, there is a transition in $\delta(q, \lambda, i, \hat{0})$ if and only if there is a transition in $\delta(q, \lambda, i, \hat{1})$ if and only if there is no transition in $\delta(q, \lambda, i, \hat{\varnothing})$. Therefore there is always at least one transition that can be applied at each step even after the end-marker, and acceptance purely depends on whether it eventually can hit a final state with all counters zero.

Lemma 14. *Given $M \in$ DTCM, a DTCM M' can be constructed in normal form such that $L(M) = L(M')$, $R(M) = R(M')$, and* runs(M) = runs(M'). *Furthermore, a* DTCM *M'' can be constructed in strong normal form such that $L(M) = L(M'')$.*

For deterministic machines, we obtain the following more nuanced situation than Proposition 12. As part of that, we see that DTCM is equivalent to 1-testable DTCM (although this increases the number of counters), which in turn in more powerful than 0-testable DTCM which is equal to DPBLIND. The proof largely uses Propositions 7, 6 and Corollary 8.

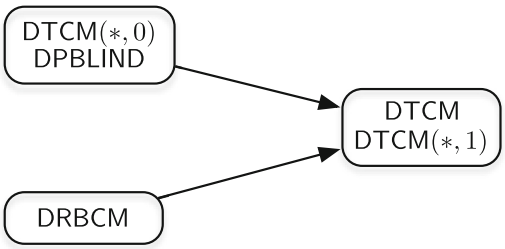

Fig. 1. In the above image, families drawn in the same cell are equal, arrows represent strict containment, and no arrows between cells represents incomparability.

Proposition 15. *The hierarchy diagram in Fig. 1 is correct.*

Despite this, we conjecture that for a fixed number of counters, there is an infinite hierarchy as r increases.

Next, we show DTCM languages are in DLOG (can be accepted in log-space by deterministic Turing machines), and hence can all be accepted by a polynomial time deterministic Turing machine. For the next result, use an encoding of DTCM whereby r is part of the description of M.

Lemma 16. *Let $M = (Q, \Sigma, \delta, q_0, F)$ be a* DTCM(k, r). *Then the counters are linearly bounded.*

From the lemma, we have:

Proposition 17. DTCM *(hence* DPBLIND*) is in* DLOG.

5 Bounded Languages in DTCM and NTCM

Next, we demonstrate that despite DTCM containing languages that are not Parikh semilinear, every bounded language in DTCM is both Ginsburg semilinear and Parikh semilinear. This result is interesting on its own, and also useful as a helper towards results in the next section. In parallel, we analyze the reachability sets and the 'runs' of DTCM accepting bounded languages. The connection between the 'runs' and bounded languages has been established in the literature. Vector addition system with states (VASS) have a concept called *flattable*, whereby a VASS is flattable [21] if the set of all runs are included in a bounded language (there, the reachability set and runs just need to be reachable from an initial configuration and not within an accepting computation). It was shown that a VASS is flattable if and only if its reachability set is definable in the Presburger arithmetic.

First we need the following two simple properties which use completely standard constructions, similar to those in [12], and therefore proofs are omitted.

Lemma 18. *For $k, r \geq 0$,* DTCM(k, r) *is closed under intersection with regular languages and inverse homomorphism.*

Next, we need another intermediate lemma, which uses the regular periodicity of the transitions applied by DTCM when accepting letter-bounded languages.

Lemma 19. *Given $\Sigma = \{a_1, \ldots, a_n\}$ and $M = (Q, \Sigma, \delta, q_0, F) \in$* DTCM$(k, r)$ *such that $L(M) \subseteq a_1^* \cdots a_n^*$, both $L(M)$ and* runs(M) *can both be accepted by an* NRBCM *and are both bounded Ginsburg and Parikh semilinear, and a representation of the semilinear sets can be effectively constructed. Moreover, $R(M) \subseteq \mathbb{N}_0^{k+1}$ is semilinear with an effective procedure.*

From these, we can obtain the following:

Proposition 20. *It is decidable, given $M \in$* DTCM(k, r), *whether $L(M)$ is bounded; and if so, we can determine words $u_1, \ldots, u_n \in \Sigma^+$ such that $L(M) \subseteq u_1^* \cdots u_n^*$, and the following are true:*

- *$L(M) \in$* NRBCM *is bounded Ginsburg semilinear, $\phi(L(M)) = Q_1 \subseteq \mathbb{N}_0^n$ is effectively semilinear;*

- runs$(M) \in$ NRBCM *is bounded Ginsburg semilinear* $\phi(\text{runs}(M)) = Q_2$ *is effectively semilinear;*
- $R(M) \subseteq \mathbb{N}_0^{k+1}$ *is effectively semilinear;*

Proof. First note that boundedness was recently shown to be decidable for NPBLIND [2,6]. Hence, given $M \in$ DTCM, we can determine whether $L(M)$ is bounded, and if so, we can determine words $u_1, \ldots, u_n \in \Sigma^+$ such that $L(M) \subseteq u_1^* \cdots u_n^*$. Henceforth, we assume $u_1, \ldots, u_n$ are known. Let $a_1, \ldots, a_n$ be new symbols and let h be a homomorphism that maps a_i to u_i for i from 1 to n. As DTCM is closed under inverse homomorphism and intersection with regular languages by Lemma 18, we can construct $M' \in$ DTCM with

$$L(M') = h^{-1}(L(M)) \cap a_1^* \cdots a_n^* = \{a_1^{i_1} \cdots a_n^{i_n} \mid u_1^{i_1} \cdots u_n^{i_n} \in L(M)\} \in \text{DTCM}.$$

By Lemma 19, $L(M)$ is bounded Ginsburg semilinear and we can effectively determine semilinear set Q such that $\phi(L(M)) = Q$. Then given $u_1, \ldots, u_n$, $\phi(L(M)) = Q$, and so $L(M)$ is in NRBCM.

Similarly, it follows that runs$(M) \in$ NRBCM by using M' with the lemma above, and also $R(M)$ is semilinear. □

If one were to examine deterministic vector addition systems with states that either operated with only λ moves, or over bounded languages, the same thing would be true.

Interestingly it was shown in [3] that the bounded Ginsburg semilinear languages are equal to the bounded languages in both NRBCM and in DRBCM. Since DRBCM $\subsetneq$ DTCM by Proposition 15, it follows that the bounded languages in DTCM and DRBCM coincide and are exactly the bounded Ginsburg semilinear languages. It is also known that the bounded languages accepted by multi-head DFAs and multi-head NFAs are also exactly the bounded Ginsburg semilinear languages (this is even the case for 2-head DFAs) [16]. And this is also true for two-way multi-head NPDA where the input heads turn at most a finite number of times. Furthermore, it follows from [3] that the bounded languages in any semilinear trio (a family closed under inverse homomorphism, λ-free homomorphism, and intersection with regular languages) are always a subset of the bounded Ginsburg semilinear languages. Some other examples are given in [3] of families of languages where the bounded languages in the family are exactly the bounded Ginsburg semilinear languages, such as finite-index ET0L, and Turing machines with a one-way input tape and a finite-turn worktape.

Corollary 21. *The bounded languages in the following families are exactly equal to the bounded Ginsburg semilinear languages:*

- DTCM,
- NRBCM,
- DRBCM,
- *multi-head* NFA,
- *2-head* DFA.

An interesting question next is whether the bounded languages in NTCM are Ginsburg or Parikh semilinear. We answer that question negatively.

Proposition 22. *The following are true:*

- *there are bounded languages in* NTCM = NPBLIND *that are not Parikh semilinear,*
- *there are bounded languages in* NTCM *that are Parikh semilinear but not Ginsburg semilinear,*
- *there are machines M in* NTCM *where $L(M)$ is bounded but $R(M)$ is not semilinear.*

Certainly though, the bounded Ginsburg semilinear languages in NTCM coincide with the bounded languages in DTCM and those of the families in Corollary 21.

To note, of all the families in Corollary 21 or that are listed above it, the only family we know of that contain languages that are not Parikh semilinear, but where the bounded languages within are only bounded Ginsburg semilinear are the multi-head DFA and NFA families, and now DTCM. It is known however that even 2-head DFA has an undecidable emptiness problem [26]. From this, it follows that it is undecidable whether a 2-head DFA accepts a bounded language (given a 2-head DFA M, construct M' to accept $L(M)\{\$\}\Sigma^*$ where $L(M) \subseteq \Sigma^*$ and \$ is a new symbol not in Σ, which is bounded if and only if $L(M) = \emptyset$). However, DTCM actually can decide if a given machine accepts a bounded language, as DTCM $\subseteq$ NPBLIND where boundedness is decidable [2,6]. Hence, DTCM is the only known class of machines with a decidable boundedness problem (which is needed for Proposition 20) that contains languages that are not Parikh semilinear, but where the bounded languages within are only Ginsburg (or Parikh) semilinear.

We obtain the following interesting property on reachability sets for NTCM. It follows from Proposition 20 that for every bounded DTCM, runs(M) is bounded, and $R(M)$ is semilinear. The following is true even for nondeterministic machines.

Corollary 23. *For each $M \in$* NTCM *(and* NPBLIND*), it is decidable if* runs(M) *is bounded; and if it is, then* runs(M) *is Ginsburg semilinear and can be accepted by a DRBCM, $R(M)$ is a semilinear set, and both semilinear sets can be effectively computed.*

Proof. Given M, build $M_\delta \in$ DTCM. Since it is decidable whether $L(M_\delta)$ is bounded [6], and if so, we can determine $x_1, \ldots, x_d$, where $x_i \in \Sigma_\delta^+$ and $L(M_\delta) \subseteq x_1^* \cdots x_d^*$; this happens if and only if runs(M) is bounded. Using Proposition 20 on $M_\delta \in$ DTCM, it then follows that $R(M_\delta)$ is semilinear, $L(M_\delta)$ is Ginsburg semilinear, and both can be effectively constructed. It is known that all bounded Ginsburg languages are in DRBCM [19]. Lemma 3 says $L(M_\delta) =$ runs(M); and if $R(M)$ were not semilinear then neither is $R(M_\delta)$. Therefore, $R(M)$ is semilinear. □

6 Complement and Containment of Deterministic Finite-Testable Machines

In this section, we show the following interesting and surprising property, that the complement of every DTCM language (hence DPBLIND) is a NTCM = NPBLIND language. Note that the machine constructed in the proof makes extremely heavy use of nondeterminism, using a nondeterministic choice at many moves of the simulation. We also conjecture that DTCM is not closed under complement (as we proved is the case with DPBLIND), but do not have a proof of this.

We require two technical lemmas which are used to decide properties of counter values that can eventually reach zeros on every counter without a zero-test. This will be helpful for constructing the complement. The first will be used after reading the end-marker to help determine if the counter values can eventually pass over a final state with all counters being zero, which is required for acceptance. This proof essentially follows from the proof in the previous section that all bounded DTCM languages are Parikh semilinear.

Lemma 24. *Given a* DTCM(k, r) $M = (Q, \Sigma, \delta, q_0, F)$ *in strong normal form with* Q *partitioned into* Q_1 *and* Q_2*, and* $q \in Q_2$*, where* $t_1, t_2, \ldots$ *is the sequence of transitions from* q *on* λ *transitions with only non-zero tests or no-tests. Then*

$$S_q = \{\vec{v}_0 \mid (p_0, \lambda, \vec{v}_0) \vdash^{t_1} \cdots \vdash^{t_l} (p_l, \lambda, \vec{v}_l), p_0 = q, \vec{v}_l = \vec{0}, p_l \in F\},$$

is an effectively determinable semilinear set.

Proof. Sequence $t_1, t_2, \ldots$ is infinite by strong normal form. S_q can be seen to be semilinear as follows: Create $M' \in$ DTCM(k, r) over $\{a_1, \ldots, a_k\}$ that on input $a_1^{j_1} \cdots a_k^{j_k} \lhd$, puts j_i on counter i and then after reading $\lhd$, simulates M. Since $L(M') \subseteq a_1^* \cdots a_k^*$, then $\psi(L(M')) = S_q$. The result is true by Lemma 19. □

We also require another technical lemma, with a proof akin to Lemma 19.

Lemma 25. *Given a* DTCM $M = (Q, \Sigma, \delta, q_0, F)$ *in strong normal form with* Q *partitioned into* Q_1 *and* Q_2*, and* $q \in Q_2$ *where* $t_1, t_2, \ldots$ *is the sequence of transitions from* q *on* λ*-transitions with only non-zero-tests or no-tests. Then*

$$R_q = \{\vec{v}_0 \mid (p_0, \lambda, \vec{v}_0) \vdash^{t_1} \cdots \vdash^{t_l} (p_l, \lambda, \vec{v}_l), p_0 = q, (t_{l+1} \text{ has non-zero-test on some counter } i \text{ and } \vec{v}_l(i) = 0) \text{ and } l \text{ is minimal where this is true}\},$$

is an effectively semilinear set.

Now we will show the main result.

Proposition 26. *For all* $L \in$ DTCM, $\overline{L} \in$ NTCM = NPBLIND.

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DTCM(k, r) in strong normal form with Q partitioned into Q_1 and Q_2. Let $q \in Q$. As in the proof of the previous lemmas, there is a unique sequence of λ transitions that can be applied starting in state q without a zero-test,

$$t_1, t_2, \ldots \tag{2}$$

This sequence could be infinite if and only if some transition t occurs in this list at least twice since M is deterministic. We can precompute whether this sequence for each $q \in Q$ is finite or infinite. Moreover, for $q \in Q_2$ the sequence must be infinite by strong normal form (see Remark 13), and we can compute the semilinear representation of

$$S_q = \{\vec{v}_0 \mid (p_0, \lambda, \vec{v}_0) \vdash^{t_1} \cdots \vdash^{t_l} (p_l, \lambda, \vec{v}_l), p_0 = q, \vec{v}_l = \vec{0}, p_l \in F, l \geq 0\}.$$

This set contains all counter values $\vec{v}_0$, such that starting in state q with $\vec{v}_0$ on the counters, it can eventually hit some ID with 0 in all counters, and in a final state. Also, we can precompute for each $q \in Q_2$ a semilinear representation of

$$R_q = \{\vec{v}_0 \mid (p_0, \lambda, \vec{v}_0) \vdash^{t_1} \cdots \vdash^{t_l} (p_l, \lambda, \vec{v}_l), p_0 = q, (t_{l+1} \text{ has a non-zero-test on counter } i \text{ and } \vec{v}_l(i) = 0) \text{ and } l \text{ is minimal where this is true}\}.$$

It is known that given two semilinear sets P_1, P_2, that $P_1 - P_2$ is also effectively semilinear [9]. As $\mathbb{N}_0^k$ is semilinear (where there is a period with a single 1 in one component and zeros in the other components), therefore $\mathbb{N}_0^k - S_q$ is effectively semilinear, which we denote by $\overline{S_q}$, and similarly for R_q producing $\overline{R_q}$.

Now we will build a NTCM $\overline{M}$ with $(2r+3)k^2$ counters accepting $\overline{L(M)}$. The machine has only a single new final state q_f. We describe $\overline{M}$ as follows. In $\overline{M}$, it operates differently before, and after the end-marker $\triangleleft$. Before the end-marker in M, the only way to not accept some input w would be if the machine crashes (a no-test transition to be applied would cause a counter to go below 0), or it enters an infinite loop on λ-transitions (note that there is always at least one transition that can be applied by strong normal form). Therefore before the end-marker (including the transition that reads $\triangleleft$ but no transition after) $\overline{M}$ simulates M faithfully using counters that we call $C_1^0, \ldots, C_k^0$, except for the following: first, before simulating each transition t of M that decreases some counter, i say, $\overline{M}$ instead makes a nondeterministic choice (we call this the *crashing strategy simulation*):

1. $\overline{M}$ guesses that the simulation can continue, and simulates t;
2. $\overline{M}$ guesses that the transition t that decreases counter i will cause counter i to go below 0. In this case, $\overline{M}$ subtracts all counters other than i by some nondeterministically guessed amount greater than or equal to 0 (but does not change counter i), and then it reads the rest of the input including $\triangleleft$, and switches to q_f (which will then accept if and only if all counters are zero). Note that here, $\overline{M}$ accepts in this way exactly when M would have crashed on t because it would have decreased counter i below zero, but in $\overline{M}$, counter i does not decrease counter i so must be zero in order for $\overline{M}$ to accept.

Second, we want $\overline{M}$ to accept if M would have entered an infinite loop on λ before hitting $\triangleleft$. After simulating each transition that reads an input symbol $a \in \Sigma$ and ending in state q, we check if the sequence (2) is infinite which was predetermined. If it is finite, we continue the simulation using the crashing strategy simulation until either it reads the next letter, or there is a zero-test in

some counter i with zero on the counter, from which $\overline{M}$ can continue to simulate. To note, if there is a successful zero-test, taking to state q', then there may be additional transitions on λ, but this will happen at most r times across an entire accepting computation for each counter.

Assume sequence (2) is infinite. Let $\alpha < \beta$ be such that $t_\alpha = t_\beta$ and they are the smallest values where this is the case and therefore the sequence is $t_1, \ldots, t_{\alpha-1}$ followed by $t_\alpha, \ldots, t_{\beta-1}$ repeated indefinitely by the determinism of M. Considering the cycle $t_\alpha, \ldots, t_{\beta-1}$, let $\vec{u} \in \mathbb{Z}^k$ be the sum of the counter values changed in this sequence. If there is some position i of this vector that is negative, then when simulating this sequence from any counter ID, it will eventually stop either by crashing, or detecting a zero with a zero-test. Then we can continue this simulation using the crashing strategy simulation. If all values in $\vec{u}$ are at least 0, then this will enter an infinite loop from a given ID if it is defined on $t_1, \ldots, t_{\beta-1}$ without a successful zero-test (if there is a successful zero-test, this cycle ends and $\overline{M}$ continues the simulation). Thus it executes this initial sequence of length $\beta - 1$ (detecting crashes with the crashing strategy simulation), and then after this sequence, M is in an infinite loop. So, $\overline{M}$ instead switches to q_f, reads the rest of the input, and reduces all counters by some nondeterministically guessed amount in order to accept.

Next, we will consider the case after reading the end-marker $\lhd$. Let $q_{(0)}$ be the state after reading $\lhd$, and let $\vec{v}_0$ be the counter values.

First, $\overline{M}$ guesses a number $m \geq 0$ such that there are m successful tests of 0 that can occur in any computation starting from $q_{(0)}$ and $\vec{v}_0$. By strong normal form $0 \leq m \leq rk$. This guessed m will later be verified. If guessed correctly, then the only way for M to not accept is after the mth successful test for zero, either M crashes, or enters an infinite loop (it does not stop by strong normal form as there is always at least one transition that can be applied), and M does not pass over any IDs with all counters 0 in a final state at any point after reading the end-marker. We will build $\overline{M}$ to guess and verify m while at the same time accepting if and only if M would not accept.

Construct $\overline{M}$ as follows: first $\overline{M}$ guesses and remembers m in the state. Then for each j one at a time from 0 to $m-1$, $\overline{M}$ makes copies of the values currently stored in counters $C_1^j, \ldots, C_k^j$ into counters named $D_1^j, \ldots, D_k^j$ and $C_1^{j+1}, \ldots, C_k^{j+1}$ respectively. Then using $D_1^j, \ldots, D_k^j$, $\overline{M}$ verifies that $\vec{v}_j \in \overline{S_{q_{(j)}}}$, thereby verifying that it will not pass over a final state with all counters zero before the next successful zero-test. To do so, it guesses a linear set in the semilinear set, then subtracts the constant, and subtracts each period a nondeterministically guessed number of times, and these counters are then verified to be 0. Then it continues to simulate M using $C_1^{j+1}, \ldots, C_k^{j+1}$ until a successful zero-test in some counter ending in some state $q_{(j+1)}$ say with counter values $\vec{v}_{j+1}$, thereby verifying that at least $j+1$ zero-tests were successful. Then it continues at the beginning of this paragraph for $j+1$ until it hits m.

When in $q_{(m)}$ with counter values $\vec{v}_m$, then it copies counters $C_1^m, \ldots, C_k^m$ into both $D_1^m, \ldots, D_k^m$ and $E_1, \ldots, E_k$. Using $E_1, \ldots, E_k$, it verifies that $\vec{v}_m \in \overline{R_{q_{(m)}}}$ (using the same technique as above where it guesses a linear set, subtracts the

constant, and each period a guessed number of times), thereby verifying that another zero-test will not be successful, and then verifies that $\vec{v}_m \in \overline{S_{q_{(m)}}}$ using $D_1^m, \ldots, D_k^m$ thereby verifying that it will not pass over a final state with all counters 0. If so, then $\overline{M}$ accepts, otherwise $\overline{M}$ will never enter a final state and cannot accept.

Next, we will verify that $\overline{L(M)} = L(\overline{M})$.

Let $w \in \overline{L(M)}$. There are several ways for M to not accept w. First, before the end-marker, M could crash by trying to subtract from a 0 counter, or it could enter an infinite loop. After the end-marker, it could not pass over all zeros in a final state, which could be the result of crashing, or entering an infinite loop without hitting all 0's in a final state.

If M crashes before the end-marker, then $\overline{M}$ would accept by guessing the exact ID before simulating the next transition causing the crash, which would therefore have to be zero before the crash. Then $\overline{M}$ can nondeterministically reduce all other counters by some amount, read the rest of the input and accept. Similarly if M gets in an infinite loop on λ before the end-marker, then in $\overline{M}$, after reading each letter of Σ or successfully simulating a zero-test, it then knows the state, and if no counter decreases in the cycle part, it can detect whether M would enter an infinite loop by executing one cycle of the loop, and so $\overline{M}$ reads the rest of the input and accepts. After reading the end-marker resulting in state $\vec{v}_0$ in state $q_{(0)}$, say that M has $0 \leq m$ successful zero-tests, where m must be less than or equal to kr (this m exists whether or not M accepts w). Then, M either crashes, or goes into an infinite loop (as mentioned earlier, it cannot stop). In any case, as w is not accepted, M will not pass over a final state with all counters being 0. If M has m successful zero-tests and does not pass over all 0 in a final state before that, then $\overline{M}$ will guess m. Indeed, after hitting each successful zero-test up to m, it is verified that from the current state $q_{(j)}$ and counter values $\vec{v}_j$, that it will not pass over a final state with all 0's in the counters by verifying that $\vec{v}_j \in \overline{S_{q_j}}$, and indeed Lemma 24 implies that it cannot pass over all 0's in a final state. If M either crashes or enters an infinite loop after the mth successful zero-test (without passing over 0's in a final state beforehand), then another successful zero-test will not occur and it will not pass over all 0's in a final state, and so $\vec{v}_m$ will be in both $\overline{S_{q_m}}$ and $\overline{R_{q_m}}$ which is enough for $\overline{M}$ to accept w. Thus, $w \in L(\overline{M})$.

Let $w \in L(\overline{M})$. Before the end-marker, $\overline{M}$ could guess that the next simulation transition that decreases some counter i would cause M to go below 0, and instead nondeterministically reduce all other counters by some amount and accept. Thus, in this situation, $w \in \overline{L(M)}$. The next way that $\overline{M}$ can accept is if there is an infinite sequence of transitions that can be applied on a non-zero-test or a no-test, and no counter value applied in a cycle can decrease, and applying this cycle at least once without a successful zero-test, which causes $\overline{M}$ to accept; in this scenario, M would then be in an infinite loop, and $w \in \overline{L(M)}$.

After the end-marker, the only way for $\overline{M}$ to accept is if it guesses a number $m \geq 0$ such that there are m successful zero-test, it can simulate M up until that mth zero-test, for each j from 0 to m, the counter values $\vec{v}_j$ and state $q_{(j)}$

right after the jth successful zero-test all have $\vec{v}_j \in \overline{S_{q_{(j)}}}$, which means that M would not pass over all counters being 0 in a final state after the jth section; and $\vec{v}_m \in \overline{R_{q_{(m)}}}$ which means that another zero-test would not be successful. Hence, M cannot accept w, and $w \in \overline{L(M)}$. □

This can be used to show the following interesting decidability property.

Theorem 27. *The containment problem is decidable for* DTCM *(and* DPBLIND*). Furthermore, the problem, "given a* NTCM M_1 *and a* DTCM M_2*, is* $L(M_1) \subseteq L(M_2)$*?" is decidable.*

Proof. Given a nondeterministic machine M_1 and a deterministic machine M_2, first construct $\overline{L(M_2)} \in$ NPBLIND $=$ NTCM by Proposition 26. Then construct $L(M_1) \cap \overline{L(M_2)} \in$ NPBLIND, as NPBLIND is closed under intersection [10], and then test emptiness [20,22], which is empty if and only if $L(M_1) \subseteq L(M_2)$. □

This result generalizes the known decidability of the containment problem for DRBCM [17] as DRBCM $\subsetneq$ DTCM.

However, it is also known that finite-crossing 2DRBCM (these are two-way DRBCMs where the input is finite-crossing (a machine is finite-crossing if every accepted word has an accepting computation where there is a bound on the number of times the boundary between each two adjacent input cells is crossed) [17], and also 2DRBCM(1) (these are two-way DRBCM machines with a single reversal-bounded counter) [15,18] have a decidable containment problem, which could be more general or incomparable to DTCM. The latter family, 2DRBCM(1) is also powerful enough to accept languages that are not Parikh semilinear like DTCM. Then, DPBLIND and DTCM join these families as having a decidable containment problem, and joins 2DRBCM(1) as one which contains languages that are not Parikh semilinear. This is quite strong as even NRBCM$(1,1)$ has an undecidable containment problem [1].

7 Future Directions

Below are some interesting problems that deserve further investigation. Although we determined here that DPBLIND is not closed under complement, it is open whether or not DTCM closed under complement. Also, is DTCM with no end-marker weaker than with the end-marker, as was the case with DPBLIND? Next, although we showed that DTCM coincides with 1-testable DTCM, is there a hierarchy in terms of DTCM(k,r) for fixed k or r? Lastly, although we showed that the complement of every DTCM is in NTCM $=$ NPBLIND, is it also true that the complement of every unambiguous NTCM (or unambiguous NPBLIND) is in NTCM?

References

1. Baker, B.S., Book, R.V.: Reversal-bounded multipushdown machines. J. Comput. Syst. Sci. **8**(3), 315–332 (1974)
2. Baumann, P., et al.: Unboundedness problems for machines with reversal-bounded counters. In: 25th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS) (2023)
3. Carpi, A., D'Alessandro, F., Ibarra, O.H., McQuillan, I.: Relationships between bounded languages, counter machines, finite-index grammars, ambiguity, and commutative regularity. Theoret. Comput. Sci. **862**, 97–118 (2021)
4. Crespi-Reghizzi, S., Pietro, P.S.: Deterministic counter machines and parallel matching computations. In: Proceedings of the 18th International Conference on Implementation and Application of Automata, CIAA 2013, vol. 7982, pp. 280–291 (2013)
5. Czerwiński, W., Hofman, P.: Language inclusion for boundedly-ambiguous vector addition systems is decidable. In: Klin, B., Lasota, S., Muscholl, A. (eds.) Proceedings of the 33rd International Conference on Concurrency Theory (CONCUR 2022), pp. 16:1–16:22 (2022)
6. Czerwinski, W., Hofman, P., Zetzsche, G.: Unboundedness problems for languages of vector addition systems. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 107, p. 119. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018)
7. Eremondi, J., Ibarra, O.H., McQuillan, I.: Insertion operations on deterministic reversal-bounded counter machines. J. Comput. Syst. Sci. **104**, 244–257 (2019)
8. Gerth, R., Peled, D., Vardi, M.Y., Wolper, P.: Simple on-the-fly automatic verification of linear temporal logic. In: PSTV 1995. IAICT, pp. 3–18. Springer, Boston, MA (1996). https://doi.org/10.1007/978-0-387-34892-6_1
9. Ginsburg, S.: The Mathematical Theory of Context-Free Languages. McGraw-Hill Inc, New York (1966)
10. Greibach, S.: Remarks on blind and partially blind one-way multicounter machines. Theoret. Comput. Sci. **7**, 311–324 (1978)
11. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston (2003)
12. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, MA (1979)
13. Hopcroft, J.E., Pansiot, J.J.: On the reachability problem for 5-dimensional vector addition systems. Theoret. Comput. Sci. **8**(2), 135–159 (1979)
14. Ibarra, O., McQuillan, I.: The effect of end-markers on counter machines and commutativity. Theoret. Comput. Sci. **627**, 71–81 (2016)
15. Ibarra, O., Yen, H.: On the containment and equivalence problems for two-way transducers. Theoret. Comput. Sci. **429**, 155–163 (2012)
16. Ibarra, O.H.: A note on semilinear sets and bounded-reversal multihead pushdown automata. Inf. Process. Lett. **3**(1), 25–28 (1974)
17. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM **25**(1), 116–133 (1978)
18. Ibarra, O.H., Jiang, T., Tran, N., Wang, H.: New decidability results concerning two-way counter machines. SIAM J. Comput. **23**(1), 123–137 (1995)

19. Ibarra, O.H., Seki, S.: Characterizations of bounded semilinear languages by one-way and two-way deterministic machines. Int. J. Found. Comput. Sci. **23**(6), 1291–1306 (2012)
20. Kosaraju, S.R.: Decidability of reachability in vector addition systems. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC 1982, pp. 267–281 (1982)
21. Leroux, J.: Presburger vector addition systems. In: Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 23–32 (2013)
22. Mayr, E.W.: An algorithm for the general Petri net reachability problem. SIAM J. Comput. **13**(3), 441–460 (1984)
23. Mottet, A., Quaas, K.: The containment problem for unambiguous register automata and unambiguous timed automata. Theory Comput. Syst. **65**, 706–735 (2021)
24. Parikh, R.: On context-free languages. J. ACM **13**(4), 570–581 (1966)
25. Pelz, E.: Closure properties of deterministic Petri nets. In: Brandenburg, F.J., Vidal-Naquet, G., Wirsing, M. (eds.) STACS 1987. LNCS, vol. 247, pp. 371–382. Springer, Heidelberg (1987). https://doi.org/10.1007/BFb0039620
26. Rosenberg, A.L.: On multi-head finite automata. In: 6th Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1965), pp. 221–228 (1965)
27. Tang, N.V.: Pushdown Automata and Inclusion Problems. Ph.D. thesis, Japan Advanced Institute of Science and Technology, Japan (2007)
28. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification. In: Chatterjee, K., Sgall, J. (eds.) Proceedings of the 1st IEEE Symposium Logic in Computer Science (LICS 1986), pp. 332–344. IEEE Computer Society (1986)
29. Vidal-Naquet, G.: Deterministic languages of Petri nets. In: Girault, C., Reisig, W. (eds.) Application and Theory of Petri Nets. Informatik-Fachberichte, vol. 52, pp. 198–202. Springer, Berlin (1981). https://doi.org/10.1007/978-3-642-68353-4_34