REGULAR SEPARABILITY OF ONE COUNTER AUTOMATA *

WOJCIECH CZERWIŃSKI AND SŁAWOMIR LASOTA

Wydział Matematyki, Informatyki i Mechaniki, University of Warsaw, Poland
e-mail address: {wczerwin,sl}@mimuw.edu.pl

ABSTRACT. The regular separability problem asks, for two given languages, if there exists a regular language including one of them but disjoint from the other. Our main result is decidability, and PSPACE-completeness, of the regular separability problem for languages of one counter automata without zero tests (also known as one counter nets). This contrasts with undecidability of the regularity problem for one counter nets, and with undecidability of the regular separability problem for one counter automata, which is our second result.

1. INTRODUCTION

We focus on separability problems for languages of finite words. We say that a language K is *separated from* another language L by a language S, if $K \subseteq S$ and $L \cap S = \varnothing$. For two families of languages $\mathcal{F}$ and $\mathcal{G}$, the $\mathcal{F}$ *separability problem for* $\mathcal{G}$ asks, for two given languages $K, L \in \mathcal{G}$ over the same alphabet, whether K is separated from L by some language from $\mathcal{F}$.

In this paper we mainly consider the separator class $\mathcal{F}$ of regular languages (thus using the term *regular separability*). As regular languages are closed under complement, K is separated from L by a regular language if, and only if, L is separated from K by a regular language. Therefore we shortly say that K *and* L are *regular separable*. As the class $\mathcal{G}$ we consider the languages of *one counter automata* (NFA extended with a non-negative counter that can be incremented, decremented and tested for zero), or its subclass — the languages of *one counter nets* (one counter automata without zero tests).

Motivation and context. Separability is a classical problem in formal languages. It was investigated most extensively for $\mathcal{G}$ the class of regular languages, and for $\mathcal{F}$ a suitable subclass thereof. Since regular languages are effectively closed under complement, the $\mathcal{F}$ separability problem is a generalization of the $\mathcal{F}$ *characterization problem*, which asks whether a given regular language L belongs to $\mathcal{F}$: indeed, $L \in \mathcal{F}$ if, and only if, L and its complement are separable by some language from $\mathcal{F}$. Separability problems for regular

languages were investigated since a long time, starting from a generic connection established by Almeida [1] between profinite semigroup theory and separability. Recently it attracted a lot of attention also outside algebraic community, which resulted in establishing the decidability of $\mathcal{F}$ separability for the family $\mathcal{F}$ of separators being, among others,

- the piecewise testable languages [11, 26]
- the locally and locally threshold testable languages [25],
- the languages definable in first order logic [28],
- the languages of certain fixed levels of the first order hierarchy [27].

The first result has been recently generalized to finite ranked trees [17].

Separability of non-regular languages attracted little attention till now. The reason for this may be twofold. First, for regular languages one can use standard algebraic tools, like syntactic monoids, and indeed most of the results have been obtained using algebraic techniques. Second, the few known negative results on separability of non-regular languages are strongly discouraging. To start off, some strong intractability results have been known already since 70's, when Szymanski and Williams proved that regular separability of context-free languages is undecidable [32]. Later Hunt [19] strengthened this result: he showed that $\mathcal{F}$ separability of context-free languages is undecidable for every class $\mathcal{F}$ containing all *definite* languages, i.e., finite Boolean combinations of languages of the form $w\Sigma^*$ for $w \in \Sigma^*$. This is a very weak condition, hence the result of Hunt suggested that nothing nontrivial can be done outside regular languages with respect to separability problems. Furthermore, Kopczyński has recently shown that regular separability is undecidable even for languages of visibly pushdown automata [22], thus strengthening the result by Szymanski and Williams once more.

On the positive side, piecewise testable separability has been shown decidable for context-free languages, languages of vector addition systems with states (VASS languages), and some other classes of languages [12]. This inspired us to start a quest for decidable cases beyond regular languages.

Once beyond regular languages, the regular separability problem seems to be the most intriguing. VASS languages is a well-known class of languages, for which the decidability status of the regular separability problem is unknown. A few positive results related to this problem have been however obtained recently. First, decidability of unary (and modular) separability of reachability sets[1] of VASS was shown in [8]; the problem is actually equivalent to regular separability of commutative closures of VASS languages. Second, decidability of regular separability of languages of Parikh automata was shown in [7]. Parikh automata recognize exactly the same languages as *integer-VASS* (a variant of VASS where one allows negative counter values [21, 5]), and therefore are a subclass of VASS languages. Finally, decidability of regular separability for *coverability* languages of VASS follows from the generic result of [10] for well-structured transition systems.

The open decidability status of regular separability of *reachability* languages of VASS is our main motivation in this paper. A more general goal is understanding for which classes of languages the regular separability problem is decidable.

Our contribution. We consider the regular separability problem for languages of one counter automata (with zero test) and its subclass, namely one counter nets (without zero test); the latter model is exactly VASS in dimension 1. The two models we call shortly

---

[1]Note that these are sets of vectors, not words.

OCA and OCN, respectively. Our main result is decidability of the regular separability problem for languages of one counter nets. Moreover, we determine the exact complexity of the problem, namely PSpace-completeness. For complexity estimations we assume a standard encoding of OCA (or OCN) and their configurations; in particular we assume binary encoding of integers appearing in the input.

Theorem 1. *Regular separability of languages of OCN is* PSpace-*complete.*

Our approach to prove decidability is by *regular over-approximation*: for every OCN language L there is a decreasing sequence of (computable) regular languages over-approximating L, such that two OCN languages are regular separable if, and only if, some pair of their approximants is disjoint. Furthermore, the latter condition can be reduced to a kind of reachability property of the synchronized product of two OCN, and effectively checked in PSpace by exploiting effective semi-linearity of the reachability set of the synchronized product. Our PSpace lower bound builds on PSpace-hardness of bounded non-emptiness of OCA [15].

It is interesting to compare the regular separability problem with the regularity problem, which asks whether a given language is regular. For every class $\mathcal{G}$ effectively closed under complement, regular separability is a generalization of regularity, as L is regular if, and only if, L and its complement $\bar{L}$ are regular separable. It turns out that regularity of OCN languages can not be reduced to regular separability: firstly because OCN languages are not closed under complement, and secondly (and more importantly) because the regularity problem is undecidable for OCN languages [33, 34] while we prove regular separability decidable.

As our second main contribution, we show that adding zero tests leads to undecidability, for any separator language class containing all definite languages. In particular, regular languages are an example of such a class.

Theorem 2. *For every language class $\mathcal{F}$ containing all definite languages, the $\mathcal{F}$ separability problem for languages of OCA is undecidable.*

Our argument is inspired by the undecidability proof by Hunt [19]: we show, roughly speaking, that every decidable problem reduces in polynomial time to the separability problem for OCA.

Organization. In Section 2 we define the models of OCA and OCN, then Sections 3–6 are devoted to the proof of Theorem 1, and finally Section 7 contains the proof of Theorem 2. The proof of Theorem 1 is factorized as follows: in Section 3 we introduce the regular over-approximation of OCN languages, in Section 4 we we provide a decision procedure for testing the disjointness property of approximants, as discussed above, further in Section 5 we provide a PSpace implementation of this procedure, and in Section 6 we give a PSpace lower bound. The last Section 8 contains some concluding remarks, including the discussion of undecidability of the regularity problem for OCN.

2. One counter automata and nets

In order to fix notation we start by recalling finite automata with ε-transitions, in a specifically chosen variant convenient when working with one counter automata.

A *nondeterministic finite automaton* (NFA) $\mathcal{A} = (Q, q_0, q_f, T)$ over a finite alphabet Σ consists of a finite set of control states Q, distinguished initial and final states $q_0, q_f \in Q$

(for convenience we assume here, w.l.o.g., a single final state), and a set of *transitions* $T \subseteq Q \times \Sigma_\varepsilon \times Q$, where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

For a word $v \in (\Sigma_\varepsilon)^*$, let $v_{|\Sigma}$ be the word obtained by removing all occurrences of ε. A run of $\mathcal{A}$ over a word $w \in \Sigma^*$ is a sequence of transitions of the form

$$(p_0, a_1, p_1), (p_1, a_2, p_2), \dots, (p_{n-1}, a_n, p_n)$$

such that $(a_1 \dots a_n)_{|\Sigma} = w$. The run is *accepting* if $p_0 = q_0$ and $p_n = q_f$. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all words w over which $\mathcal{A}$ has an accepting run. Languages of NFA are called *regular*.

One counter automata and nets. In brief, a one counter automaton (OCA) is an NFA with a non-negative counter, where we allow for arbitrary changes of the counter value in one step.

Formally, an OCA is a tuple $\mathcal{A} = (Q, \alpha_0, \alpha_f, T, T_{=0})$, where Q are control states as above. A *configuration* $(q, n) \in Q \times \mathbb{N}$ of $\mathcal{A}$ consists of a control state and a non-negative counter value. There are two distinguished configurations, the initial one $\alpha_0 = (q_0, n_0)$ and the final one $\alpha_f = (q_f, n_f)$. The finite set $T \subseteq Q \times \Sigma_\varepsilon \times Q \times \mathbb{Z}$ contains *transitions* of $\mathcal{A}$. A transition (q, a, q', z) can be fired in a configuration $\alpha = (q, n)$ if $n + z \geqslant 0$, leading to a new configuration $\alpha' = (q', n + z)$. We write $\alpha \xrightarrow{a} \alpha'$ if this is the case. Finally, the set $T_{=0} \subseteq Q \times \Sigma_\varepsilon \times Q$ contains *zero tests*. A zero test (q, a, q') can be fired in a configuration $\alpha = (q, n)$ only if $n = 0$, leading to a new configuration $\alpha' = (q', n)$. Again, we write $\alpha \xrightarrow{a} \alpha'$ if this is the case.

A run of an OCA over a word $w \in \Sigma^*$ is a sequence of transitions and zero tests of the form

$$\alpha_0 \xrightarrow{a_1} \alpha_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} \alpha_n$$

such that $(a_1 \dots a_n)_{|\Sigma} = w$; we briefly write $\alpha_0 \xrightarrow{w} \alpha_n$ if this is the case, and $\alpha_0 \longrightarrow \alpha_n$ if a word w is irrelevant. The run is *accepting* if α_0 is the initial configuration of $\mathcal{A}$, and α_n is the final one. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all words w over which $\mathcal{A}$ has an accepting run.

A one counter net (OCN) is an OCA without zero tests, i.e., one with $T_{=0} = \varnothing$. We drop the component $T_{=0}$ and denote OCN as $(Q, \alpha_0, \alpha_f, T)$. In other words, an OCN is exactly a vector addition system with states (VASS) in dimension 1 [20, 18].

Example 3. *Consider two OCN languages over the alphabet $\{a, b\}$:*

$$K = \{a^n b^n \mid n \in \mathbb{N}\} \qquad L = \{a^n b^{n+1} \mid n \in \mathbb{N}\}.$$

An example regular language separating K from L is $R = \{a^n b^m \mid n \equiv m \mod 2\}$. Indeed, R includes K and is disjoint with L. On the other hand, K and $L' = \{a^n b^m \mid m > n\}$ are not regular separable (which follows from Corollary 10 below).

Other modes of acceptance. We briefly discuss other possible modes of acceptance of OCA.

First, consider a variant of OCA with a finite set of initial configurations, and a finite set of final ones. This variant can be easily simulated by OCA as defined above. Indeed, add two fresh states q_0, q_f, and fix the initial and final configurations $\alpha_0 = (q_0, 0)$ and $\alpha_f = (q_f, 0)$. Moreover, add ε-transitions enabling to go from α_0 to every of former initial configurations, and symmetrically add ε-transitions enabling to go from every of former final configurations to α_f.

The above simulation reveals that w.l.o.g. we can assume that the counter values n_0 and n_f in the initial and final configurations are 0. This will be implicitly assumed in the rest of the paper.

Yet another possibility is accepting solely by control state: instead of a final configuration $\alpha_f = (q_f, n_f)$, such an OCA would have solely a final control state q_f, and every run ending in a configuration (q_f, n), for arbitrary n, would be considered accepting. Again, this variant is easily simulated by our model: it is enough to assume w.l.o.g. that q_f has no outgoing transitions nor zero tests, add a transition $(q_f, \varepsilon, q_f, -1)$ decrementing the counter in the final state, and fix the final configuration as $(q_f, 0)$.

Finally, note that all the simulations discussed above work for OCN as well. In particular, in the sequel we may assume, w.l.o.g., that the counter values in initial and final configurations of OCN are 0.

3. Regular over-approximation of OCN

For an OCN $\mathcal{A}$ and $n > 0$, we are going to define an NFA $\mathcal{A}_n$ which we call *n-approximation* of $\mathcal{A}$. As the main result of this section we prove (as Corollary 10) that two OCN (and even OCA as pointed out in Remark 11) are regular separable if, and only if, their n-approximations are disjoint for some $n > 0$.

As long as the counter value is below n, the automaton $\mathcal{A}_n$ stores this value exactly (we say then that $\mathcal{A}_n$ is in *low* mode); if the counter value exceeds n, the automaton $\mathcal{A}_n$ only stores the remainder of the counter value modulo n (we say then that $\mathcal{A}_n$ is in *high* mode). Thus $\mathcal{A}_n$ can pass from low mode to high one; but $\mathcal{A}_n$ can also nondeterministically decide to pass the other way around, from high to low mode.

Let Q be the state space of $\mathcal{A}$, and let $(q_0, 0)$ and $(q_f, 0)$ be its initial and final configurations. As the state space of $\mathcal{A}_n$ we take the set

$$Q_n = Q \times \{0, \ldots, n-1\} \times \{\text{LOW}, \text{HIGH}\}.$$

The initial and final state of $\mathcal{A}_n$ are $(q_0, 0, \text{LOW})$ and $(q_f, 0, \text{LOW})$, respectively. Every transition (q, a, q', z) of $\mathcal{A}$ induces a number of transitions of $\mathcal{A}_n$, as defined below (for any c satisfying $0 \leqslant c < n$):

$$\big((q, c, \text{LOW}), a, (q', c+z, \text{LOW})\big) \qquad\qquad \text{if } 0 \leqslant c+z < n$$
$$\big((q, c, \text{LOW}), a, (q', (c+z) \bmod n, \text{HIGH})\big) \qquad\qquad \text{if } n \leqslant c+z$$
$$\big((q, c, \text{HIGH}), a, (q', (c+z) \bmod n, \text{LOW})\big) \qquad\qquad \text{if } c+z < 0$$
$$\big((q, c, \text{HIGH}), a, (q', (c+z) \bmod n, \text{HIGH})\big).$$

Note that passing from high mode to low one is only possible if the counter value (modulo n) drops, after an update, strictly below 0; in particular, this requires $z < 0$.

Example 4. *Recall the languages K and L from Example 3, and consider an OCN $\mathcal{A}$ recognizing K that has two states q_0, q_f, and three transitions:*

$$(q_0, a, q_0, +1)$$
$$(q_0, \varepsilon, q_f, 0)$$
$$(q_f, b, q_f, -1).$$

The 2-approximating automaton $\mathcal{A}_2$ has 8 states $\{q_0, q_f\} \times \{0, 1\} \times \{\text{LOW}, \text{HIGH}\}$. In state $(q_0, 1, \text{LOW})$ on letter a, the automaton is forced to change the mode to HIGH; symmetrically, in state $(q_f, 0, \text{HIGH})$ on letter b, the automaton can change its mode back to LOW:

$$\big((q_0, 1, \text{LOW}), a, (q_0, 0, \text{HIGH})\big)$$
$$\big((q_f, 0, \text{HIGH}), b, (q_f, 1, \text{LOW})\big).$$

Otherwise, the mode is preserved by transitions; for instance, in high mode the automaton changes the state irrespectively of the input letter: for every $q \in \{q_0, q_f\}$, $x \in \{a, b\}$ and $c \in \{0, 1\}$, there is a transition

$$\big((q, c, \text{HIGH}), x, (q, 1 - c, \text{HIGH})\big).$$

The language recognized by $\mathcal{A}_2$ is

$$\{a^n b^m \mid (n = m < 2) \vee (n, m \geqslant 2 \wedge n \equiv m \ mod \ 2)\}.$$

According to the definition above, the automaton $\mathcal{A}_n$ can oscillate between low and high mode arbitrarily many times. Actually, as we argue below, it is enough to allow for at most one oscillation.

Proposition 5. *For every run of $\mathcal{A}_n$ between two states in high mode, there is a run over the same word between the same states which never exits the high mode.*

Proof. Indeed, observe that if $\mathcal{A}_n$ has any of the following transitions

$$\big((q, m, \text{LOW}), a, (q', m', \text{LOW})\big)$$
$$\big((q, m, \text{LOW}), a, (q', m', \text{HIGH})\big)$$
$$\big((q, m, \text{HIGH}), a, (q', m', \text{LOW})\big)$$

then $\mathcal{A}_n$ necessarily has also the transition

$$\big((q, m, \text{HIGH}), a, (q', m' \bmod n, \text{HIGH})\big).$$

Thus every run oscillating through high and low modes that starts and ends in high mode, can be simulated by a one that never exits high mode. □

A run of an OCN $\mathcal{A}$ we call *n-low*, if the counter value is strictly below n in all configurations of the run. Proposition 6 below characterizes the language of $\mathcal{A}_n$ in terms of runs of $\mathcal{A}$, and will be useful for proving the Approximation Lemma below. Then Corollary 7, its direct consequence, summarizes some properties of approximation useful in the sequel.

Proposition 6. *Let $\mathcal{A} = (Q, (q_0, 0), (q_f, 0), T)$ be an OCN, and let $n > 0$. Then $w \in L(\mathcal{A}_n)$ if, and only if,*
(a) *either $\mathcal{A}$ has an n-low accepting run over w,*
(b) *or w factorizes into $w = w_{\text{PREF}} w_{\text{MID}} w_{\text{SUFF}}$, such that $\mathcal{A}$ has the following runs*

$$\begin{aligned} (q_0, 0) &\xrightarrow{w_{\text{PREF}}} (q, n + d) \\ (q, cn + d) &\xrightarrow{w_{\text{MID}}} (q', c'n + d') \\ (q', n + d') &\xrightarrow{w_{\text{SUFF}}} (q_f, 0), \end{aligned} \tag{3.1}$$

for some states $q, q' \in Q$ and natural numbers $c, c' \geqslant 1$ and $d, d' \geqslant 0$.

Proof. We start with the "if" direction. If there is an n-low run over w in $\mathcal{A}$ then clearly $w \in L(\mathcal{A}_n)$. Otherwise, suppose that $w = w_{\text{PREF}} w_{\text{MID}} w_{\text{SUFF}}$ and the words w_{PREF}, w_{MID} and w_{SUFF} admit the runs as stated in (3.1) above. Then clearly $\mathcal{A}_n$ admit the following runs:

$$(q_0, 0, \text{LOW}) \xrightarrow{w_{\text{PREF}}} (q, d \bmod n, \text{HIGH})$$

$$(q, d \bmod n, \text{HIGH}) \xrightarrow{w_{\text{MID}}} (q', d' \bmod n, \text{HIGH})$$

$$(q', d' \bmod n, \text{HIGH}) \xrightarrow{w_{\text{SUFF}}} (q_f, 0, \text{LOW})$$

and thus $(q_0, 0) \xrightarrow{w} (q_f, 0)$ in $\mathcal{A}_n$ as required.

For the "only if" direction, suppose $w \in L(\mathcal{A}_n)$. If $\mathcal{A}_n$ has a run over w that never exits low mode, then clearly $\mathcal{A}$ has an n-low run over w. Otherwise, consider any run of $\mathcal{A}_n$ over w. Distinguish the first and the last configuration in high mode along this run, say (q, d, HIGH) and (q', d', HIGH). The two configurations determine a factorization of the word w into three parts $w = w_{\text{PREF}} w_{\text{MID}} w_{\text{SUFF}}$ such that $\mathcal{A}_n$ admit the following runs:

$$(q_0, 0, \text{LOW}) \xrightarrow{w_{\text{PREF}}} (q, d, \text{HIGH})$$

$$(q, d, \text{HIGH}) \xrightarrow{w_{\text{MID}}} (q', d', \text{HIGH})$$

$$(q', d', \text{HIGH}) \xrightarrow{w_{\text{SUFF}}} (q_f, 0, \text{LOW}).$$

The first and the last run imply the first and the last run in (3.1). For the middle one, we may assume (w.l.o.g., by Proposition 5) that $\mathcal{A}_n$ never exits high mode, which implies immediately existence of the middle run in (3.1). □

Corollary 7. *Let $\mathcal{A}$ be an OCN and let $m, n > 0$. Then*
(a) $L(\mathcal{A}) \subseteq L(\mathcal{A}_n)$,
(b) $L(\mathcal{A}_n) \subseteq L(\mathcal{A}_m)$ *if $m \mid n$.*

Proof. The first inclusions follow easily by the characterization of Proposition 6. The second one is easily shown by definition of n-approximation. □

Now we state and prove the Approximation Lemma, which is the crucial property of approximation. In the sequel we will strongly rely on direct consequences of this lemma, formulated as Corollaries 9 and 10 below.

Lemma 8 (Approximation Lemma). *For an OCN $\mathcal{A}$, the following conditions are equivalent:*
(a) $L(\mathcal{A})$ *is empty,*
(b) $L(\mathcal{A}_n)$ *is empty, for some $n > 0$.*

Proof. Clearly (b) implies (a), by Corollary 7(a). In order to prove that (a) implies (b), fix $\mathcal{A} = (Q, (q_0, 0), (q_f, 0), T)$ and suppose that the languages $L(\mathcal{A}_n)$ are *non-empty* for *all* $n > 0$; our aim is to show that $L(\mathcal{A})$ is non-empty as well.

In the sequel we do not need the non-emptiness assumption for *all* n; it will be enough to use the assumption for some fixed n computed as follows. Let $|Q|$ be the number of states of $\mathcal{A}$ and $d_{\mathcal{A}}$ be the maximal absolute value of integer constants appearing in transitions T of $\mathcal{A}$. Then let $K = |Q| \cdot d_{\mathcal{A}}$, and let $n = K!$ ($K!$ stands for K factorial.)

Let w be a fixed word that belongs to $L(\mathcal{A}_n)$. Our aim is to produce a word w' that belongs to $L(\mathcal{A})$, by a pumping in the word w; the pumping will allow to make a run of $\mathcal{A}_n$ into a correct run of $\mathcal{A}$.

As $w \in L(\mathcal{A}_n)$, by Proposition 6 we learn that w satisfies one of conditions (a), (b). If w satisfies (a) then $w' = w \in L(\mathcal{A})$ as required. We thus concentrate, from now on, on the

case when w satisfies condition (b) in Proposition 6. Let's focus on the first (fixed from now on) run of $\mathcal{A}$ in (3.1), namely

$$(q_0, 0) \xrightarrow{w_{\text{PREF}}} (q, n + d),$$

for some prefix w_{PREF} of w and $d \geqslant 0$. This run starts with the counter value 0, and ends with the counter value at least n. We are going to analyze closely the prefix of the run that ends immediately before the counter value exceeds K for the first time; denote this prefix by ρ. A configuration (q, m) in ρ we call *latest* if the counter value stays strictly above m in all the following configurations in ρ. In other words, a latest configuration is the last one in ρ where the counter value is at most m. A crucial but easy observation is that the difference of counter values of two consecutive latest configurations is at most $d_{\mathcal{A}}$. Therefore, as K has been chosen large enough, ρ must contain more than $|Q|$ latest configurations. By the pigeonhole principle, there must be a state of $\mathcal{A}$, say q, that appears in at least two latest configurations. In consequence, for some infix v of w_{PREF}, the OCN $\mathcal{A}$ has a run over v of the form

$$(q, m) \xrightarrow{v} (q, m'), \quad \text{for some } m < m' \leqslant m + K.$$

As a consequence, the word v can be repeated an arbitrary number of times, preserving correctness of the run but increasing the final counter value. Recall that the final counter value of ρ is $n + d$, while we would like to achieve $cn + d$ (for c in Proposition 6). Modify the word w_{PREF} by adding $(c - 1) \cdot n / (m' - m)$ repetitions of the word v, thus obtaining a new word w'_{PREF} such that $\mathcal{A}$ has a run

$$(q_0, 0) \xrightarrow{w'_{\text{PREF}}} (q, cn + d). \tag{3.2}$$

In exactly the same way we modify the suffix w_{SUFF} of w, thus obtaining a word w'_{SUFF} over which the OCN $\mathcal{A}$ has a run

$$(q', c'n + d') \xrightarrow{w'_{\text{SUFF}}} (q_f, 0). \tag{3.3}$$

By concatenation we obtain a word $w' = w'_{\text{PREF}} w_{\text{MID}} w'_{\text{SUFF}}$ which is accepted by $\mathcal{A}$, by composition of the run (3.2), the middle run in (3.1), and the run (3.3). Thus $L(\mathcal{A})$ is non-empty, as required. $\qquad\square$

As OCNs are closed under products with finite automata and these products commute with n-approximations, we get:

Corollary 9. *For an OCN $\mathcal{A}$ and a regular language R, the following conditions are equivalent:*

(a) *$L(\mathcal{A})$ and R are disjoint,*
(b) *$L(\mathcal{A}_n)$ and R are disjoint, for some $n > 0$.*

Proof. Fix an OCN $\mathcal{A} = (Q, (q_0, 0), (q_f, 0), T)$ and an NFA $\mathcal{B} = (P, p_0, p_f, U)$ recognizing the language R. For convenience we assume here that $\mathcal{A}$ has an ε-transition of the form $(q, \varepsilon, q, 0)$ in every state $q \in Q$, and $\mathcal{B}$ has a self-loop ε-transitions (p, ε, p) in every state $p \in P$. We will use the synchronized product $\mathcal{A} \times \mathcal{B}$ of OCN $\mathcal{A}$ and NFA $\mathcal{B}$, which is the OCN defined by

$$\mathcal{A} \times \mathcal{B} = \big(Q \times P, ((q_0, p_0), 0), ((q_f, p_f), 0), V\big),$$

where a transition $\big((q, p), a, (q', p'), z\big) \in V$ if $(q, a, q', z) \in T$ and $(p, a, p') \in U$. Observe that the synchronized product construction commutes with n-approximation: up to isomorphism

of finite automata,

$$(\mathcal{A} \times \mathcal{B})_n \;=\; \mathcal{A}_n \times \mathcal{B}. \tag{3.4}$$

Condition (a) in Corollary 9 is equivalent to emptiness of the product $\mathcal{A} \times \mathcal{B}$ which, by the Approximation Lemma applied to $A \times \mathcal{B}$, is equivalent to emptiness of the *left* automaton in (3.4), for some n. Therefore condition (a) is also equivalent to emptiness of the *right* automaton in (3.4), for some n. Finally, the latter condition is equivalent to condition (b). □

Corollary 10. *For two OCN $\mathcal{A}$ and $\mathcal{B}$, the following conditions are equivalent:*

(a) $L(\mathcal{A})$ *and* $L(\mathcal{B})$ *are regular separable,*
(b) $L(\mathcal{A}_n)$ *and* $L(\mathcal{B})$ *are disjoint, for some $n > 0$,*
(c) $L(\mathcal{A}_n)$ *and* $L(\mathcal{B}_n)$ *are disjoint, for some $n > 0$.*

Proof. In order to prove that (a) implies (b), suppose that a regular language R separates $L(\mathcal{B})$ from $L(\mathcal{A})$, i.e., R includes $L(\mathcal{B})$ and is disjoint from $L(\mathcal{A})$. By Corollary 9 we learn that for some $n > 0$, R and $\mathcal{A}_n$ are disjoint. Thus necessarily $L(\mathcal{B})$ and $L(\mathcal{A}_n)$ are disjoint too.

To show that (b) implies (c) use Corollary 9 for OCN $\mathcal{B}$ and regular language $L(\mathcal{A}_n)$. We get that there exists $m > 0$ such that $L(\mathcal{B}_m)$ and $L(\mathcal{A}_n)$ are disjoint. Then using Corollary 7(b) we have that $L(\mathcal{A}_{nm})$ and $L(\mathcal{B}_{nm})$ are disjoint as well.

Finally, (c) easily implies (a), as any of the regular languages $L(\mathcal{A}_n)$, $L(\mathcal{B}_n)$ can serve as a separator (Corollary 7(a) is used here). □

At this stage we do not have yet any effective bound on the minimal n satisfying condition (b) or (c) in Corollary 10. Our decision procedure for OCN, to be presented in the next section, will test condition (b). A bound on n in Corollary 10(b)–(c) can be extracted from the decision procedure (as discussed in Section 5); however, this bound does not lead directly to the optimal PSpace complexity.

Remark 11. Interestingly, exactly the same notion of approximation can be defined for OCA as well. Even if Propositions 5 and 6 are no more valid for OCA, all other facts proved in this section still hold for this more general model, in particular the Approximation Lemma and Corollaries 9 and 10. Confronting this with undecidability of regular separability for OCA (which we prove in Section 7) leads to a conclusion that the characterizations of Corollary 10 are not effectively testable in case of OCA, while they are in case of OCN.

4. Decision procedure

Our proof of PSpace-membership of the regular separability problem for OCN splits into two parts. In this section we do the first step: we reduce the (non-)separability problem of two OCN $\mathcal{A}$ and $\mathcal{B}$ to a kind of reachability property in the synchronized product of $\mathcal{A}$ and $\mathcal{B}$, and then build the decision procedure relying on semi-linearity of the corresponding reachability relation. As the second (more technical) step, in Section 5 we concentrate on implementing the decision procedure in PSpace: we encode the reachability property using (multiple) systems of linear Diophantine equations which will be all enumerable (and hence solvable) in polynomial space.

Semi-linear sets. For a set $P \subseteq \mathbb{Z}^l$ of vectors, let $P^* \subseteq \mathbb{Z}^l$ contain all vectors that can be obtained as a finite sum, possibly the empty one, and possibly with repetitions, of vectors from P. In other words, P^* is the set of *non-negative* linear combinations of vectors from P. *Linear sets* are sets of the form $L = \{b\} + P^*$, where $b \in \mathbb{Z}^l$, P is a finite subset of $\mathbb{Z}^l$, and addition $+$ is understood element-wise. Thus L contains sums of the vector b and a vector from P^*. The vector b is called *base*, and vectors in P *periods*; we write shortly $b + P^*$. Finite unions of linear sets are called *semi-linear*. We use sometimes a special case of semi-linear sets of the form $B + P^*$, for finite sets B, P.

We will often consider (semi-)linear sets with some coordinates non-negative, e.g., $b + P^* \subseteq \mathbb{N}^2 \times \mathbb{Z}$. Note that in this case we necessarily have $b \in \mathbb{N}^2 \times \mathbb{Z}$ and $P \subseteq \mathbb{N}^2 \times \mathbb{Z}$.

Vector addition systems with states. We start by recalling the notion of *integer* vector addition systems with states (integer-VASS). For $d > 0$, a d-dimensional integer-VASS $\mathcal{V} = (Q, T)$, or d-integer-VASS, consists of a finite set Q of control states, and a finite set of transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$. A configuration of $\mathcal{V}$ is a pair $(q, v) \in Q \times \mathbb{Z}^d$ consisting of a state and an integer vector. Note that we thus allow, in general, negative values in configuration (this makes a difference between integer-VASS and VASS); however later we will typically impose non-negativeness constraints on a selected subset of coordinates. A d-integer-VASS $\mathcal{V}$ determines a step relation between configurations: there is a step from (q, v) to (q', v') if T contains a transition (q, z, q') such that $v' = v + z$. We write $(q, v) \longrightarrow (q', v')$ if there is a sequence of steps leading from (q, v) to (q', v'), and say that (q', v') is *reachable* from (q, v) in $\mathcal{V}$.

Synchronized product. For convenience we assume that every OCN has an ε-transition of the form $(q, \varepsilon, q, 0)$ in every control state q. We will use a synchronized product operation over one counter nets. For two OCN $\mathcal{A} = (Q, \alpha_0, \alpha_f, T)$ an $\mathcal{B} = (P, \beta_0, \beta_f, U)$, their *synchronized product* $\mathcal{A} \otimes \mathcal{B}$ is a 2-integer-VASS whose states are pairs of states $Q \times P$ of $\mathcal{A}$ and $\mathcal{B}$, respectively, and whose transitions contain all triples

$$\big((q, p), (z, v), (q', p')\big)$$

such that there exists $a \in \Sigma_\varepsilon$ with $(q, a, q', z) \in T$ and $(p, a, p', v) \in U$. Note that $\mathcal{A} \otimes \mathcal{B}$ is unlabeled — the alphabet letters are only used to synchronize $\mathcal{A}$ and $\mathcal{B}$ — and allows, contrarily to $\mathcal{A}$ and $\mathcal{B}$, for negative values on both coordinates. Moreover note that there is no distinguished initial or final configuration in an integer-VASS.

We will later need to impose a selective non-negativeness constraint on values of configurations. For a d-integer-VASS $\mathcal{V}$ and a sequence $C_1, \ldots, C_d$, where $C_i = \mathbb{N}$ or $C_i = \mathbb{Z}$ for each i, by $\mathcal{V}(C_1, \ldots, C_d)$ we mean the transition system of $\mathcal{V}$ truncated to the subset $Q \times C_1 \times \cdots \times C_d \subseteq Q \times \mathbb{Z}^d$ of configurations. For instance, $(\mathcal{A} \otimes \mathcal{B})(\mathbb{N}, \mathbb{N})$ differs from $\mathcal{A} \otimes \mathcal{B}$ by imposing the non-negativeness constraint on both coordinates, and is thus a 2-VASS. On the other hand, in $(\mathcal{A} \otimes \mathcal{B})(\mathbb{Z}, \mathbb{N})$ the counter of $\mathcal{A}$ can get arbitrary integer values while the counter of $\mathcal{B}$ is restricted to be non-negative.

Disjointness assumption. Fix, for this and the next section, two input OCN

$$\mathcal{A} = (Q, (q_0, 0), (q_f, 0), T) \quad \text{and} \quad \mathcal{B} = (P, (p_0, 0), (p_f, 0), U),$$

and let $\mathcal{V} = \mathcal{A} \otimes \mathcal{B}$ be their synchronized product. If the intersection of $L(\mathcal{A})$ and $L(\mathcal{B})$ is non-empty, the answer to the separability question is obviously negative. We may thus consider only input OCN $\mathcal{A}$ and $\mathcal{B}$ with $L(\mathcal{A})$ and $L(\mathcal{B})$ are disjoint. This is eligible as the

disjointness can be effectively checked in PSPACE. Indeed, the intersection of $L(\mathcal{A})$ and $L(\mathcal{B})$ is nonempty if, and only if,

$$\big((q_0, p_0), 0, 0\big) \longrightarrow \big((q_f, p_f), 0, 0\big)$$

in the 2-VASS $\mathcal{V}(\mathbb{N}, \mathbb{N})$, which can be checked in PSPACE by the result of [4].

Assumption 12. *For the decision procedure we assume, w.l.o.g., that $L(\mathcal{A}) \cap L(\mathcal{B}) = \varnothing$.*

Reduction to a reachability property of $\mathcal{A} \otimes \mathcal{B}$. Recall Corollary 10(b) which characterizes regular non-separability by non-emptiness of the intersection of $L(\mathcal{A}_n)$ and $L(\mathcal{B})$, for all $n > 0$, which, roughly speaking, is equivalent to a reachability property in the synchronized product of NFA $\mathcal{A}_n$ and the OCN $\mathcal{B}$, for all $n > 0$. We are going now to internalize the quantification over all n, by transferring the reachability property to the synchronized product $\mathcal{V} = \mathcal{A} \otimes \mathcal{B}$ of the two OCN $\mathcal{A}$ and $\mathcal{B}$.

For convenience we introduce the following terminology. For $n > 0$ we say that $\mathcal{V}$ *admits n-reachability* (or *n*-reachability holds in $\mathcal{V}$) if there are $q, q' \in Q$, $p, p' \in P$, $m, m' \geqslant n$, $l, l' \geqslant 0$ and $x \in \mathbb{Z}$ such that $n | x$ and the following conditions hold:

$$\mathrm{PREF}_q^p(m, l) : \quad \big((q_0, p_0), 0, 0\big) \longrightarrow \big((q, p), m, l\big) \text{ in } \mathcal{V}(\mathbb{N}, \mathbb{N}),$$

$$\mathrm{MID}_{qq'}^{pp'}(m, l, m' + x, l') : \quad \big((q, p), m, l\big) \longrightarrow \big((q', p'), m' + x, l'\big) \text{ in } \mathcal{V}(\mathbb{Z}, \mathbb{N}),$$

$$\mathrm{SUFF}_{q'}^{p'}(m', l') : \quad \big((q', p'), m', l'\big) \longrightarrow \big((q_f, p_f), 0, 0\big) \text{ in } \mathcal{V}(\mathbb{N}, \mathbb{N}).$$

The *n*-reachability in $\mathcal{V}$ differs in three respects from ordinary reachability $\big((q_0, p_0), 0, 0\big) \longrightarrow \big((q_f, p_f), 0, 0\big)$ in $\mathcal{V}(\mathbb{N}, \mathbb{N})$. First, we require two intermediate values of the counter in $\mathcal{A}$, namely m, m', to be at least n. Second, in the middle part we allow the counter of $\mathcal{A}$ to be negative. Finally, we allow for a mismatch $x \in \mathbb{Z}$ between the the middle and the final part, as long as x is divisible by n. Thus *n*-reachability does *not* imply non-emptiness $(q_0, 0) \longrightarrow (q_f, 0)$ of $\mathcal{A}$. On the other hand, *n*-reachability *does* imply non-emptiness $(p_0, 0) \longrightarrow (p_f, 0)$ of $\mathcal{B}$.

Proposition 13. *$\mathcal{A}$ and $\mathcal{B}$ are not regular separable if, and only if, $\mathcal{V}$ admits n-reachability for all $n > 0$.*

Proof. Using the characterization of Corollary 10(b), it suffices to show that for every $n > 0$, $L(\mathcal{A}_n) \cap L(\mathcal{B}) \neq \varnothing$ if, and only if, $\mathcal{V}$ admits *n*-reachability. Fix $n > 0$ in the sequel.

For the "only if" direction, let $w \in L(\mathcal{A}_n) \cap L(\mathcal{B})$. As $w \in L(\mathcal{A}_n)$, we may apply Proposition 6. Note that the condition (a) of Proposition 6 surely does not hold, as $w \notin L(\mathcal{A})$ due to the disjointness assumption; therefore condition (b) must hold for some states $q, q' \in Q$ and natural numbers $c, c' \geqslant 1$ and $d, d' \geqslant 0$. Put $m := n + d$, $m' := n + d'$ and $x := (c' - c + 1)n$ (recall that $m' + x$ may be negative). As $w \in L(\mathcal{B})$, the corresponding states p, p' and counter values l, l' can be taken from the corresponding two positions in an accepting run of $\mathcal{B}$ over w. The chosen states q, q', p, p' and integer values m, m', l, l', x prove *n*-reachability in $\mathcal{V}$, as required.

For the "if" direction suppose that $\mathcal{V}$ admits n-reachability, and let w_{PREF}, w_{MID} and w_{SUFF} be some words witnessing the three conditions of n-reachability:

$$((q_0, p_0), 0, 0) \xrightarrow{w_{\text{PREF}}} ((q, p), m, l) \text{ in } \mathcal{V}(\mathbb{N}, \mathbb{N}),$$

$$((q, p), m, l) \xrightarrow{w_{\text{MID}}} ((q', p'), m' + x, l') \text{ in } \mathcal{V}(\mathbb{Z}, \mathbb{N}),$$

$$((q', p'), m', l') \xrightarrow{w_{\text{SUFF}}} ((q_f, p_f), 0, 0) \text{ in } \mathcal{V}(\mathbb{N}, \mathbb{N}),$$

for some $q, q' \in Q$, $p, p' \in P$, $m, m' \geqslant n$, $l, l' \geqslant 0$ and $x \in \mathbb{Z}$. In particular, this implies

$$(q, m + (c-1)n) \xrightarrow{w_{\text{MID}}} (q', m' + x + (c-1)n) \text{ in } \mathcal{A} \tag{4.1}$$

for $c \geqslant 1$ large enough. This also implies that the word $w = w_{\text{PREF}} w_{\text{MID}} w_{\text{SUFF}}$ belongs to $L(\mathcal{B})$. We will prove that w also belongs to $L(\mathcal{A}_n)$, by demonstrating that the factorization $w = w_{\text{PREF}} w_{\text{MID}} w_{\text{SUFF}}$ satisfies the condition (b) in Proposition 6. Indeed, for $d := m - n$, $d' := m' - n$, we obtain then runs over m_{PREF} and m_{SUFF} as required in (b) in Proposition 6. In order to get a run over w_{MID}, we take $c \geqslant 1$ large enough so that (4.1) holds; for $c' := c + x/n$, (4.1) rewrites, as required, to $(q, cn + d) \xrightarrow{w_{\text{MID}}} (q', c'n + d')$ in $\mathcal{A}$. $\qquad\square$

Witnesses and effective semi-linearity. Building on Proposition 13, we are going to design a decision procedure to check whether $\mathcal{V}$ admits n-reachability for all $n > 0$. To this end we use the three conditions of n-reachability as subsets of pairs resp. quadruples of integers,

$$\text{PREF}_q^p, \ \text{SUFF}_q^p \subseteq \mathbb{N}^2, \qquad \text{MID}_{qq'}^{pp'} \subseteq \mathbb{N}^2 \times \mathbb{Z} \times \mathbb{N}, \tag{4.2}$$

and define the set $\mathcal{R} \subseteq \mathbb{N}^2 \times \mathbb{Z}$ by the following formula with three variables (m, m', x):

$$\exists q, q' \in Q, \ p, p' \in P, \ l, l' \in \mathbb{N}, \ \text{PREF}_q^p(m, l) \wedge \text{MID}_{qq'}^{pp'}(m, l, m'+x, l') \wedge \text{SUFF}_{q'}^{p'}(m', l'). \tag{4.3}$$

Then n-reachability is equivalent to saying that some $(m, m', x) \in \mathcal{R}$ satisfies

$$m, m' \geqslant n \quad \text{and} \quad n | x. \tag{4.4}$$

Any triple $(m, m', x) \in \mathcal{R}$ satisfying the condition (4.4) we call n-*witness* in the sequel. In this terminology, our algorithm is to decide whether $\mathcal{R}$ contains n-witnesses for all $n > 0$.

Proposition 14. *The set $\mathcal{R}$ is effectively semi-linear, i.e., a union of linear sets*

$$\mathcal{R} = L_1 \cup \ldots \cup L_k, \tag{4.5}$$

where $L_i = b_i + P_i^*$ for effectively computable bases $b_i \in \mathbb{N}^2 \times \mathbb{Z}$ and periods $P_i \subseteq_{fin} \mathbb{N}^2 \times Z$ ($i = 1, \ldots, k$).

Proof. All the sets in (4.2) are effectively semi-linear. Indeed, PREF_q^p is essentially the reachability set of a 2-VASS, and thus effectively semi-linear [4, 18], and likewise for SUFF_q^p. Moreover, effective semi-linearity of $\text{MID}_{qq'}^{pp'}$ can be derived directly from Parikh's theorem; specifically, it follows from Lemma 3.4 in [14] (in Section 5 an explicit proof is provided, giving additionally an exponential bound on representation size of the semi-linear set). In consequence, as semi-linear sets are effectively closed under boolean combinations and projections, and the ternary relation of addition is semi-linear, the set $\mathcal{R}$ is effectively semi-linear too. $\qquad\square$

The next lemma allows us to consider each of the linear sets separately:

Lemma 15. *If a finite union $X_1 \cup \ldots \cup X_k \subseteq \mathbb{N}^2 \times \mathbb{Z}$ contains n-witnesses for all $n > 0$, then some of $X_1, \ldots, X_k$ also does.*

Proof. We use a monotonicity property: if $n'|n$ then every n-witness is automatically also n'-witness. Consider a sequence of $(n!)$-witnesses, for $n > 0$, contained in X. One of the sets $X_1, \ldots, X_k$ necessarily contains infinitely many of them. By monotonicity, this set contains $(n!)$-witnesses for all $n > 0$, and hence n-witnesses for all $n > 0$. $\qquad\square$

Decision procedure. Relying on Proposition 14 and Lemma 15, our procedure enumerates the linear sets (4.5) and chooses one of them. It thus remains to solve the core problem: given $b \in \mathbb{N}^2 \times \mathbb{Z}$ and $P \subseteq_{\text{fin}} \mathbb{N}^2 \times \mathbb{Z}$, decide whether $L = b + P^*$ contains n-witnesses for all $n > 0$. For such sets L, the condition we are to check boils down to two separate sub-conditions:

Lemma 16. *$L = b + P^*$ contains n-witnesses for all $n > 0$ if, and only if,*
(a) *for every n, there is $(m, m', x) \in L$ with $m, m' \geqslant n$; and*
(b) *for every n, there is $(m, m', x) \in L$ with $n|x$.*

Proof. Put $b = (b_1, b_2, b_3)$. Indeed, if $(b_1, b_2, b_3) + (k_1, k_2, k_3) \in L$ for $b_1 + k_1, b_2 + k_2 \geqslant n$, and $(b_1, b_2, b_3) + (m_1, m_2, m_3) \in L$ for $n|(b_3 + m_3)$, then $(b_1, b_2, b_3) + n(k_1, k_2, k_3) + (m_1, m_2, m_3) \in L$ is an n-witness. Hence conditions (a) and (b) imply that L contains n-witnesses for all $n > 0$. The opposite direction is obvious. $\qquad\square$

Since all vectors in P are non-negative on the first two coordinates, condition (a) in Lemma 16 is easy for algorithmic verification: enumerate vectors in P while checking whether some vector is positive on first coordinate, and some (possibly different) vector is positive on second coordinate.

As the last bit of our decision procedure, it remains to check condition (b) in Lemma 16. Writing b_3, resp. P_3, for the projection of b, resp. P, on the third coordinate, we need to check whether the set $b_3 + P_3^* \subseteq \mathbb{Z}$ contains (possibly negative) multiplicities of all $n > 0$. We build on:

Lemma 17. *The set $b_3 + P_3^*$ contains multiplicities of all $n > 0$ if, and only if, b_3 is a linear combination of P_3, i.e.,*

$$b_3 = a_1 p_1 + \cdots + a_k p_k, \tag{4.6}$$

for $a_1, \ldots, a_k \in \mathbb{Z}$ and $p_1, \ldots, p_k \in P_3$.

Proof. For the "only if" direction, suppose that $b_3 + P_3^*$ contains multiplicities of all positive numbers. If $b_3 = 0$ then it is the empty linear combination of P_3; suppose therefore that $b_3 \neq 0$. Note that this implies in particular that P_3 is forcedly nonempty. Fix an arbitrary $n \in P_3$. Suppose $n > 0$ (if $n < 0$ take $-n$ instead of n). By the assumption, $b_3 + p \equiv 0 \bmod n$ for some $p \in P_3^*$, i.e.,

$$b_3 \equiv -p \bmod n.$$

Then $b_3 = -p + an$ for some $a \in \mathbb{Z}$, hence a linear combination of P_3 as required.

For the "if" direction, suppose b_3 is a linear combination of P_3 as in (4.6), and let $n > 0$. It is possible to decrease the numbers $a_1, \ldots, a_k$ by multiplicities of n so that they become non-positive. Thus we have

$$b_3 \equiv (a_1 - c_1 n)p_1 + \cdots + (a_k - c_k n)p_k \bmod n,$$

for $a_1 - c_1 n \leqslant 0, \ldots, a_k - c_k n \leqslant 0$, i.e., $b_3 \equiv -p \bmod n$ for $p = (c_1 n - a_1)p_1 + \cdots + (c_k n - a_k)p_k \in P_3{}^*$. In consequence $b_3 + p \equiv 0 \bmod n$, as required. $\qquad\square$

Thus we only need to check whether b_3 is a linear combination of P_3. By the Chinese remainder theorem, this is equivalent to b_3 being a multiplicity of the greatest common divisor of all numbers in P_3. Thus our decision procedure enumerates the set P, computes the greatest common divisor g of projections p_3 on the third coordinate of all vectors $p \in P$, and finally checks whether $g|b_3$. This completes the description of the decision procedure.

5. PSpace upper bound

In this section we prove the PSpace upper bound of Theorem 1. All the PSpace complexity statements below are understood with respect to the size of the two input OCN, under binary encoding of integers.

As before, fix two OCN $\mathcal{A}$ and $\mathcal{B}$ with disjoint languages. The decision procedure from Section 4 *enumerates* all linear sets $L = b + P^*$ appearing in an effectively computed representation of the semi-linear set $\mathcal{R}$. For obtaining the upper bound we need to provide suitable estimations on representation size of these linear sets. To this aim we introduce the concept of PSpace-*enumerable sets*, whose semi-linear representation can be effectively enumerated in polynomial space.

PSpace-enumerable sets. For a finite set of vectors $P \subseteq_{\mathrm{fin}} \mathbb{Z}^l$, we say that an algorithm *enumerates* P if it computes consecutive elements of a sequence $p_1, \ldots, p_m$, possibly with repetitions, such that $P = \{p_1, \ldots, p_m\}$; in other words, every element of P appears at least once in the sequence, but no other element does. An algorithm enumerates a linear set $L = b + P^* \subseteq \mathbb{Z}^l$ if it first computes b and then enumerates P. If there is a polynomial space algorithm which enumerates $L = b + P^*$, the set L is called PSpace-*enumerable*. A semi-linear set S we call PSpace-enumerable (slightly abusing the notation) if for some sequence of linear sets $L_1, \ldots, L_k$ such that

$$S = L_1 \cup \ldots \cup L_k,$$

there is a polynomial space algorithm that first enumerates L_1, then enumerates L_2, and so on, and finally enumerates L_k. In particular, this means that for some polynomial bound N, every base and every period can be stored using at most N bits.

PSpace upper bound. Propositions 18 and 19 below state that all the sets appearing in (4.2) are PSpace-enumerable. Their direct consequence, Proposition 20, says the same about the set $\mathcal{R}$, and forms the cornerstone of PSpace decision procedure.

Proposition 18. *For every $q \in Q$ and $p \in P$, the semi-linear sets PREF_q^p and SUFF_q^p are* PSpace-*enumerable.*

Proposition 19. *For every $q, q' \in Q$ and $p, p' \in P$, the semi-linear set $\mathrm{MID}_{qq'}^{pp'}$ is* PSpace-*enumerable.*

Proposition 20. *The set $\mathcal{R}$ is* PSpace-*enumerable.*

Before proving Propositions 18–20, we notice that Proposition 20 allows us to implement in polynomial space the enumeration of the semi-linear set $\mathcal{R}$ which underlies the decision procedure presented in Section 4. This yields the upper bound of Theorem 1.

Bound on the size of separator. A further consequence of Proposition 20 is a bound on the minimal value of n in Corollary 10(b), which naturally leads to a complexity upper bound. Indeed, due to Proposition 20 such a bound can be extracted from the proofs of Lemmas 15–17 but, importantly, it is only *doubly exponential* (hence, exhaustive checking if $L(\mathcal{A}_n) \cap L(\mathcal{B}) \neq \varnothing$ for all n so bounded would only yield an EXPSPACE algorithm). First, the proof of Lemma 17 reveals that if the set $b_3 + P_3{}^*$ does not contain multiplicities of all $n > 0$, and P_3 is nonempty, then it does not contain multiplicities of some period $n \in P_3$, hence n is bounded exponentially (one can obtain an even better bound for n, namely the greatest common divisor of all periods from P_3, which still is only exponentially bounded in general). Thus by Lemma 16 we get an exponential bound on the smallest n such that a linear set L does not contain an n-witness. Then we lift the bound to semi-linear sets (cf. Lemma 15) but at the price of increasing it to doubly exponential: indeed, if every component linear set L_i does not contain an n_i-witness for some $n_i > 0$, the semi-linear set $\mathcal{R} = L_1 \cup \ldots \cup L_k$ does not contain an n-witness, for n the least common multiplicity of all n_i and hence doubly exponential in general. We have thus bounded the smallest n such that n-reachability does not hold in $\mathcal{V}$, and consequently (cf. the proof of Proposition 13) n in Corollary 10(b). We do not know if this bound can be improved to single exponential (which would immediately make the exhaustive check a PSPACE algorithm). The question seems to be quite challenging, as the proofs of Propositions 18–20 rely on a combination of several nontrivial results [2, 4, 23].

The rest of Section 5 is devoted to the proofs of Propositions 18–20.

5.1. **Proof of Proposition 18.**

We concentrate on showing that the sets PREF_q^p are PSPACE-enumerable. (The sets SUFF_q^p can be dealt with in exactly the same way as PREF_q^p, but with $\mathcal{V}$ replaced by the reverse of $\mathcal{V}$.) In the sequel fix states q, p of $\mathcal{A}$ and $\mathcal{B}$, respectively. The set PREF_q^p is nothing but the reachability set of a 2-VASS $\mathcal{V}(\mathbb{N}, \mathbb{N})$ in control state (q, p), from the initial configuration $((q_0, p_0), 0, 0)$. We build on a result of [4] which describes the reachability set in terms of sets reachable via a finite set of *linear path schemes*, a notion that we are going to recall now.

Let T be set of transitions of $\mathcal{V}$. A linear path scheme is a regular expression over T of the form:

$$E = \alpha_0 \beta_1^* \alpha_1 \ldots \beta_k^* \alpha_k, \tag{5.1}$$

where $\alpha_i, \beta_i \in T^*$. The sequences $\beta_1, \ldots, \beta_k$ are called *loops* of E. By *length* of E we mean the sum of lengths of all α_i and β_i. Let REACH_E (the reachability set via E) contain all pairs $(n, m) \in \mathbb{N}^2$ such that $((q_0, p_0), 0, 0) \longrightarrow ((q, p), n, m)$ in $\mathcal{V}(\mathbb{N}, \mathbb{N})$ via a sequence of transitions that belongs to E.

Here is Thm. 1 in [4], translated to our terminology:

Lemma 21 [4]. *There are computable bounds N_1, N_2, where N_1 is exponential and N_2 is polynomial in the size of $\mathcal{V}$, such that PREF_q^p is the union of sets REACH_E, for linear path schemes E of length at most N_1, with at most N_2 loops.*

Our decision procedure, instead of dealing explicitly with a linear path scheme E, will rather use $4k + 2$ pairs of integers characterizing E, as described above. Let $a_i \in \mathbb{Z}^2$, for $i = 0, \ldots, k$, be the total effect of executing the sequence α_i, i.e., a_i is the sum of effects of consecutive transitions in α_i, and likewise b_i for the sequence β_i, for $i = 1, \ldots, k$. Moreover, let $c_i \in \mathbb{N}^2$, for $i = 0, \ldots, k$ be the (point-wise) minimal nonnegative values of counters

that allow to execute the sequence α_i (in $\mathcal{V}(\mathbb{N}, \mathbb{N})$), and likewise d_i for the sequence β_i, for $i = 1, \ldots, k$. The $4k + 2$ pairs of numbers, namely a_i, c_i (for $i = 0, \ldots, k$) and b_i, d_i (for $i = 1, \ldots, k$), we jointly call the *profile* of the linear path scheme E.

Lemma 22. *Given pairs $a_i \in \mathbb{Z}^2, c_i \in \mathbb{N}^2$ (for $i = 0, \ldots, k$) and $b_i \in \mathbb{Z}^2, d_i \in \mathbb{N}^2$ (for $i = 1, \ldots, k$), it can be checked in* PSpace *if they form the profile of some linear path scheme.*

Proof. Guess intermediate control states $(q_1, p_1), \ldots, (q_k, p_k)$ and put $(q_{k+1}, p_{k+1}) = (q, p)$. Check that the following reachability properties hold in $\mathcal{V}(\mathbb{N}, \mathbb{N})$, for $i = 0, \ldots, k$ and $i = 1, \ldots, k$, respecively:

$$\big((q_i, p_i), c_i\big) \longrightarrow \big((q_{i+1}, p_{i+1}), c_i + a_i\big)$$
$$\big((q_i, p_i), d_i\big) \longrightarrow \big((q_i, p_i), d_i + b_i\big),$$

and that the above properties fail to hold if any c_i (resp. d_i) is replaced by a point-wise smaller pair of numbers. All the required checks are instances of the reachability problem for 2-VASS, hence doable in PSpace [4]. $\qquad\square$

Denote by Reach_p the set of configurations reachable in $\mathcal{V}(\mathbb{N}, \mathbb{N})$ via some linear path scheme with profile p. Using Lemma 22 we can enumerate all profiles of linear path schemes (5.1) of length at most N_1 with $k \leq N_2$ loops. Note that each such profile can be represented (in binary) in polynomial space: as N_1 is exponential and N_2 is polynomial, the profile is defined by polynomially many numbers a_i, b_i, c_i, d_i, each of them at most exponential. Thus by the virtue of Lemma 21 it is enough to show, for a fixed profile p, that the set Reach_p is PSpace-enumerable. Fix a profile p from now on.

As a convenient tool we will use *linear Diophantine equations*. These are systems of equations of the form

$$a_1 x_1 + \cdots + a_l x_l = a, \tag{5.2}$$

where $x_1, \ldots, x_l$ are variables, and $a, a_1, \ldots, a_l$ are integer coefficients. For a system $\mathcal{U}$ of such equations, we denote by $\text{sol}(\mathcal{U}) \subseteq \mathbb{N}^l$ the solution set of $\mathcal{U}$, i.e., the set all of non-negative integer vectors $(n_1, \ldots, n_l)$ such that the valuation $x_1 \mapsto n_1, \ldots, x_l \mapsto n_l$ satisfies all the equations in $\mathcal{U}$.

We say that a vector is *bounded* by m if it is smaller than m on every coordinate. From the results of [13, 29] (see also Prop. 2 in [6] for the convenient formulation) we get a bound on the size of semi-linear representation of the set of solutions of $\mathcal{U}$ which is polynomial in the number of variables and values of coefficients of $\mathcal{U}$, but exponential in the number of equations:

Lemma 23 [13, 6, 29]**.** *Let $\mathcal{U}$ be a system of d linear Diophantine equations with n variables such that the absolute values of coefficients are bounded by M. Then $\text{sol}(\mathcal{U}) = B + P^*$, with every base $b \in B$ and period $p \in P$ bounded by $\mathcal{O}(n \cdot M)^d$.*

Observe that, forcedly, $P \subseteq \text{sol}(\mathcal{U}_0)$ where $\mathcal{U}_0$ denotes a modification of the system of linear equations $\mathcal{U}$ with all right-hand side constants a (cf. (5.2)) replaced by 0. We will use Lemma 23 once we state the last lemma we need (the idea is based on [4]):

Lemma 24. *The set Reach_p is a projection of the union*

$$\text{sol}(\mathcal{U}^1) \cup \ldots \cup \text{sol}(\mathcal{U}^l),$$

for systems of linear Diophantine equations $\mathcal{U}^1 \dots \mathcal{U}^l$ that can be enumerated in polynomial space.

Proof. Recall that $c_0 \in \mathbb{N}^2$ denotes the (point-wise) minimal nonnegative values of counters that allow to execute α_0 in $\mathcal{V}(\mathbb{N}, \mathbb{N})$. We assume that $c_0 = (0,0)$, as otherwise the set REACH_p is empty. Introduce variables x_i, for $i = 1 \dots k$, to represent the number of times the loop β_i has been executed. The necessary and sufficient condition for executing the linear path scheme (5.1) can be described by a positive Boolean combination of linear inequalities (that can be further transformed into linear equations using auxiliary variables), which implies Lemma 24.

Indeed, for every $i = 1, \dots, k$, we distinguish two sub-cases, $x_i = 0$ or $x_i > 0$. In the former case (the loop β_i is *not* executed) we put the two inequalities described succinctly as (note that $a_j, b_j \in \mathbb{Z}^2$, while $c_j, d_j \in \mathbb{N}^2$ for all j)

$$a_0 + b_1 x_1 + a_1 + \cdots + b_{i-1} x_{i-1} + a_{i-1} \geqslant c_i$$

to say that α_i can be executed. In the latter case (the loop β_i *is* executed) we put the following four inequalities

$$a_0 + b_1 x_1 + a_1 + \cdots + b_{i-1} x_{i-1} + a_{i-1} \geqslant d_i$$
$$a_0 + b_1 x_1 + a_1 + \cdots + b_{i-1} x_{i-1} + a_{i-1} + b_i(x_i - 1) \geqslant d_i$$

to say that the first and the last iteration of the loop β_i can be executed (which implies that each intermediate iteration of β_i can be executed as well), plus the two inequalities

$$a_0 + b_1 x_1 + a_1 + \cdots + b_{i-1} x_{i-1} + a_{i-1} + b_i x_i \geqslant c_i$$

to assure that α_i can be executed next. Finally, in both cases the two variables (y_1, y_2) representing the final configuration, are related to other variables by the two equations:

$$a_0 + b_1 x_1 + a_1 + \cdots + b_k x_k + a_k = (y_1, y_2).$$

Thus for every sequence of choices between $x_i = 0$ and $x_i > 0$, for $i = 1, \dots, k$, we obtain a system of equations. The set REACH_p is the projection, onto (y_1, y_2), of the union of solution sets of all these systems of equations. $\qquad\square$

The last two lemmas immediately imply that REACH_p is PSPACE-enumerable. Indeed, by Lemma 23 applied to every of the systems $\mathcal{U}^i$, we have $\text{sol}(\mathcal{U}^i) = B_i + P_i^*$ for bases B_i containing all vectors $b \in \text{sol}(\mathcal{U}^i)$ bounded by N, and periods P_i containing all vectors $p \in \text{sol}(\mathcal{U}^i{}_0)$ bounded by N, where N is exponential and computable. Relying on Lemma 24, the algorithm enumerates all systems $\mathcal{U}^i$, then enumerates all $b \in B_i$ satisfying the above constraints, and for each b it enumerates all periods $p \in P_i$ satisfying the above constraints. The proof of Proposition 18 is thus completed.

5.2. Proof of Proposition 19.
In the sequel we fix states q, q' of $\mathcal{A}$ and p, p' of $\mathcal{B}$, respectively. Our aim is to prove that $\text{MID}_{qq'}^{pp'}$ is PSPACE-enumerable, by encoding this set as Parikh image of an OCN.

Recall that Parikh image $\text{PI}(w)$ of a word $w \in \Sigma^*$, for a fixed ordering $a_1 < \cdots < a_k$ of Σ, is defined as the vector $(n_1, \dots, n_k)$ where n_i is the number of occurrences of a_i in w, for $i = 1, \dots, k$. Parikh image lifts to languages: $\text{PI}(L) = \{\text{PI}(w) \mid w \in L\}$.

An OCN we call *1-OCN* if all its transitions (q, a, q', z) satisfy $z \in \{-1, 0, 1\}$. We define a 1-OCN $\mathcal{C}$ of exponential size, over a 5-letter alphabet $\{a_0, b_0, a_+, a_-, b_f\}$, such that $\text{MID}_{qq'}^{pp'}$

is the image of a linear function of $\text{PI}(L(\mathcal{C}))$. $\mathcal{C}$ starts with the zero counter value, and its execution splits into three phases. In the first phase $\mathcal{C}$ reads arbitrarily many times a_0 without modifying the counter, and arbitrary many times b_0, increasing the counter by 1 at every b_0. Thus the counter value of $\mathcal{C}$ at the end of the first phase is equal to the number of b_0s.

In the last phase, $\mathcal{C}$ reads arbitrarily many times b_f, decreasing the counter by 1 at every b_f. The accepting configuration of $\mathcal{C}$ requires the counter to be 0. Thus the counter value of $\mathcal{C}$ at the beginning of the last phase must be equal to the number of b_fs.

In the intermediate phase $\mathcal{C}$ simulates the execution of $\mathcal{V}(\mathbb{Z}, \mathbb{N})$. The counter value of $\mathcal{C}$ corresponds, during this phase, to the counter value of $\mathcal{B}$. On the other hand, the counter value of $\mathcal{A}$ will only be reflected by the number of a_+ and a_- read by $\mathcal{C}$. States of $\mathcal{C}$ correspond to pairs of states of $\mathcal{A}$ and $\mathcal{B}$, respectively; there will be also exponentially many auxiliary states. The phase starts in state (q, p), and ends in state (q', p'). A transition $((q_1, p_1), (z_1, z_2), (q_2, p_2))$ of $\mathcal{V}$ is simulated in $\mathcal{C}$ as follows: First, if $z_1 \geqslant 0$ then $\mathcal{C}$ reads z_1 letters a_+; otherwise, $\mathcal{C}$ reads $-z_1$ letters a_-. Second, if $z_2 \geqslant 0$ then $\mathcal{C}$ performs z_2 consecutive increments of the counter; otherwise $\mathcal{C}$ performs $-z_2$ decrements. In both tasks, fresh auxiliary states are used. We assume w.l.o.g. that every transition of $\mathcal{V}$ satisfies $(z_1, z_2) \neq (0, 0)$; hence $\mathcal{C}$ has no ε-transitions. This completes the description of the 1-OCN $\mathcal{C}$.

Let $S = \text{PI}(L(\mathcal{C})) \subseteq \mathbb{N}^5$. Then $\text{MID}_{qq'}^{pp'} = f(S)$, for the linear function $f : \mathbb{Z}^5 \to \mathbb{Z}^4$ defined by (intensionally, we re-use alphabet letters in the role of variable names):

$$(a_0, b_0, a_+, a_-, b_f) \mapsto (a_0, b_0, a_0 + a_+ - a_-, b_f).$$

Therefore if S is PSpace-enumerable then $f(S)$ is also so; it thus remains to prove that S is PSpace-enumerable.

Our proof builds on results of [2, 23]. In order to state it we need to introduce the concept of *pump* of an accepting run ρ of $\mathcal{C}$ (called *direction* in [2]). We treat accepting runs ρ as sequences of transitions. A pump of ρ of the first kind is a sequence α of transitions such that ρ factorizes into $\rho = \rho_1 \rho_2$, and $\rho_1 \alpha \rho_2$ is again an accepting run. Note that in this case the effect of α on the counter is necessarily 0. A pump of second kind is a pair α, β of sequences of transitions, where the effect of α is non-negative, such that ρ factorizes into $\rho = \rho_1 \rho_2 \rho_3$, and $\rho_1 \alpha \rho_2 \beta \rho_3$ is again an accepting run. Note that in this case the effect of β is necessarily opposite to the effect of α.

Parikh image of a sequence of transitions $\text{PI}(\rho)$ is understood as a shorthand for Parikh image of the input word of ρ. Furthermore, we use a shorthand notation for Parikh image of a pump π: let $\text{PI}(\pi)$ mean either $\text{PI}(\alpha)$ or $\text{PI}(\alpha\beta)$, in case of the first or second kind, respectively. Similarly, the length of π is either the length of α, or the length of $\alpha\beta$. Lemma 25 is a direct consequence of Lemma 55 in [3] (see also Lemma 15 in [2, 3], or Theorem 6 in [23]):

Lemma 25. *There is a computable bound N, polynomial in $size(\mathcal{C})$, such that S is a union of linear sets of the form*

$$\text{PI}(\rho) + \{\text{PI}(\pi_1), \ldots, \text{PI}(\pi_l)\}^* \quad (l \leqslant 5),$$

where ρ is an accepting run of $\mathcal{C}$ of length at most N, and $\pi_1 \ldots \pi_l$ are pumps of ρ of length at most N.

Except for different (but equivalent) notation, Lemma 55 in [3] differs from Lemma 25 only by claiming that a run ρ determines uniquely pumps $\pi_1, \ldots, \pi_l$ (which is not needed here).

We need one more fact, in which we refer to the computable bound N of Lemma 25 (note that $\mathcal{O}(\log \mathrm{size}(C))$ is polynomial in the sizes of $\mathcal{A}$ and $\mathcal{B}$, as $\mathcal{C}$ is blown up exponentially compared to $\mathcal{A}$ and $\mathcal{B}$):

Lemma 26. *For given $b \in \mathbb{N}^5$ and $P = \{p_1, \ldots, p_l\} \subseteq \mathbb{N}^5$, $l \leqslant 5$, it is decidable in space $\mathcal{O}(\log \mathrm{size}(C))$ if there is an accepting run ρ of $\mathcal{C}$ of length at most N and pumps $\pi_1, \ldots, \pi_l$ of ρ of length at most N, such that $b = \mathrm{PI}(\rho)$ and $p_i = \mathrm{PI}(\pi_i)$ for $i = 1, \ldots, l$.*

Proof. As the first step, the algorithm guesses, for each $i = 1, \ldots, l$, whether pump π_i would be of first or second kind. For simplicity of presentation we assume below that all pumps are guessed to be of second kind, i.e., they are pairs α_i, β_i – pumps of first kind are treated in a simpler but similar way. Expanding the definitions, the algorithm is to check if there is an accepting run ρ and pumps α_i, β_i ($i = 1, \ldots, l$), all of length at most N and of desired Parikh image, so that for every i the run factors into

$$\rho \;=\; \rho_i^1 \, \rho_i^2 \, \rho_i^3 \tag{5.3}$$

and the following sequence is again an accepting run:

$$\rho_i \;=\; \rho_i^1 \, \alpha_i \, \rho_i^2 \, \beta_i \, \rho_i^3.$$

To this aim the algorithm simulates in parallel, using space $\mathcal{O}(\log \mathrm{size}(C))$, at most $l + 1 \leqslant 6$ different runs of $\mathcal{C}$, one of them corresponding to ρ and the remaining l ones corresponding to $\rho_1, \ldots, \rho_l$.

Since N is polynomial in $\mathrm{size}(\mathcal{C})$, counting up to N is possible in space $\mathcal{O}(\log \mathrm{size}(C))$. The algorithm starts by simulating nondeterministically a run ρ of $\mathcal{C}$ of length at most N. We call this simulation *main thread* (in addition, there will be also at most l auxiliary *pump threads*, as introduced below). The algorithm thus maintains the current configuration c of $\mathcal{C}$ in the main thread, and chooses nondeterministically consecutive transitions of $\mathcal{C}$ to execute. In addition, the algorithm maintains Parikh image $a \in \mathbb{N}^l$ of the run executed so far, updated after every step of simulation and stored in space $\mathcal{O}(\log \mathrm{size}(C))$. In the course of simulation the algorithm guesses nondeterministically l positions corresponding to the ends of prefixes ρ_i^1 in (5.3), for $i = 1, \ldots, l$. At each so guessed position all the threads are suspended, and a new *pump* thread is added. The new thread is responsible for simulating, from the current configuration c of $\mathcal{C}$, some sequence of transitions α_i of length at most N. The simulation of α_i finishes nondeterministically, say in configuration c_i, if the counter value of c_i is greater or equal to the counter value of c (the effect of α_i is non-negative) and the control state of c_i is the same as the control state of c. Then Parikh image $a_i = \mathrm{PI}(\alpha_i)$ is stored (in space $\mathcal{O}(\log \mathrm{size}(C))$), and the simulation of all threads (the suspended ones and the new one) are continued, with the proviso that all threads use *the same* nondeterministically chosen sequence of transitions. Up to now, the algorithm maintains up to $l + 1$ configurations $c, c_1, \ldots, c_l$ of $\mathcal{C}$, and stores up to $l + 1$ vectors $a, a_1 \ldots a_l \in \mathbb{N}^l$.

Later in the course of simulation the algorithm also guesses nondeterministically l positions in ρ, corresponding to the the beginnings of suffixes ρ_i^3 in (5.3), for $i = 1, \ldots, l$. Similarly as above, at each so guessed position all threads are suspended except for the one corresponding to pump i, and that pump thread simulates some sequence of transitions β_i of length at most N. This simulation terminates only if its current configuration becomes equal to the current configuration of the main thread, i.e., $c_i = c$, and moreover Parikh image of β_i, say b_i, satisfies $a_i + b_i = p_i$; and once this happens, the pump thread corresponding to pump i is cancelled. Note that the pump threads are not necessarily synchronized, and it might happen that one pump thread becomes cancelled even before some another pump

thread even starts. This lack of synchronization is clearly not an issue, as the total number of simultaneously executed pump threads stays below l. The whole simulation finishes when all pump threads are cancelled, the current configuration c is the final configuration of $\mathcal{C}$, and Parikh image of the run executed in the main thread equals b. ▢

The last two lemmas imply that S is PSpace-enumerable. Indeed, it is enough to enumerate all candidates b, P bounded by N, as specified in Lemma 25, and validate them, using Lemma 26. This completes the proof of Proposition 19.

5.3. **Proof of Proposition 20.** Recall the definition (4.3) of the set $\mathcal{R}$ as a projection of a conjunction of constraints:

$$\exists q, q' \in Q,\, p, p' \in P,\, l, l' \in \mathbb{N},\,\, \mathrm{PREF}_q^p(m, l) \wedge \mathrm{MID}_{qq'}^{pp'}(m, l, m'+x, l') \wedge \mathrm{SUFF}_{q'}^{p'}(m', l').$$

We are going to prove that $\mathcal{R}$ is PSpace-enumerable. The algorithm enumerates quadruples of states q, q', p, p'. For each fixed quadruple, it enumerates (by Propositions 18 and 19) component linear sets of PREF_q^p, $\mathrm{MID}_{qq'}^{pp'}$ and SUFF_q^p. Thus it is enough to consider three fixed PSpace-enumerable linear sets

$$L_{\mathrm{PREF}} = b_{\mathrm{PREF}} + P_{\mathrm{PREF}}{}^* \subseteq \mathrm{PREF}_q^p \subseteq \mathbb{N}^2$$

$$L_{\mathrm{MID}} = b_{\mathrm{MID}} + P_{\mathrm{MID}}{}^* \subseteq \mathrm{MID}_{qq'}^{pp'} \subseteq \mathbb{N}^2 \times \mathbb{Z} \times \mathbb{N}$$

$$L_{\mathrm{SUFF}} = b_{\mathrm{SUFF}} + P_{\mathrm{SUFF}}{}^* \subseteq \mathrm{SUFF}_q^p \subseteq \mathbb{N}^2.$$

We now treat each of these linear sets as a system of linear Diophantine equations. For instance, if $P_{\mathrm{PREF}} = \{p_1, \ldots, p_k\}$, all pairs $(x, y) \in L_{\mathrm{PREF}}$ are described by two equations

$$(x_{\mathrm{PREF}}, y_{\mathrm{PREF}}) = b_{\mathrm{PREF}} + p_1 x_1 + \cdots + p_k x_k, \tag{5.4}$$

for fresh variables $x_1, \ldots, x_k$. Note that the number of variables is exponential, but the number of equations is constant, namely equal two. The same can be done with two other linear sets L_{MID} and L_{SUFF}, yielding 6 more equations involving 6 variables $x_{\mathrm{MID}}, y_{\mathrm{MID}}, x'_{\mathrm{MID}}, y'_{\mathrm{MID}}, x_{\mathrm{SUFF}}, y_{\mathrm{SUFF}}$, plus exponentially many other fresh variables. Points in L_{SUFF} are represented as $(x_{\mathrm{SUFF}}, y_{\mathrm{SUFF}})$, while points in L_{MID} as $(x_{\mathrm{MID}}, y_{\mathrm{MID}}, x'_{\mathrm{MID}}, y'_{\mathrm{MID}})$. (Since we only consider non-negative solutions of systems of equations, in case of L_{MID} two separate cases should be considered, depending on whether the final value x'_{MID} is non-negative or not. For simplicity we only consider here the case when it is non-negative.) In addition, we add the following equations (and one variable x):

$$x_{\mathrm{PREF}} = x_{\mathrm{MID}}$$

$$y_{\mathrm{PREF}} = y_{\mathrm{MID}}$$

$$y'_{\mathrm{MID}} = y_{\mathrm{SUFF}}$$

$$x = x'_{\mathrm{MID}} - x_{\mathrm{SUFF}}.$$

In total, we have a system $\mathcal{U}$ of 12 equations (8 mentioned before and 4 presented above) involving exponentially many variables, including these 9 variables

$$x_{\mathrm{PREF}}, y_{\mathrm{PREF}}, x_{\mathrm{MID}}, y_{\mathrm{MID}}, x'_{\mathrm{MID}}, y'_{\mathrm{MID}}, x_{\mathrm{SUFF}}, y_{\mathrm{SUFF}}, x. \tag{5.5}$$

The value of 3 of them is relevant for us, namely $x_{\mathrm{PREF}}, x_{\mathrm{SUFF}}$ and x. We claim that the projection of the solution set $\mathrm{sol}(\mathcal{U})$ onto these 3 *relevant* coordinates is PSpace-enumerable. To prove this we apply once more Lemma 23, to deduce that the solution set of $\mathcal{U}$ is of the

form $B + P^*$, for all bases $b \in B$ and all periods $p \in P$ bounded exponentially. Hence each number appearing in a base or a period is representable in polynomial space; the same applies to the projection onto the 3 relevant coordinates. Finally, observe that the projections of the sets B and P onto the 3 relevant coordinates can be enumerated in polynomial space. Indeed, in order to check whether a vector in $\mathbb{N}^3$ is (the projection of) a B or P, we proceed similarly as in the proof of Proposition 18: the algorithm first guesses values of other 6 variables among (5.5), and then guesses the values of other (exponentially many) variables on-line, in the course of enumerating the coefficients p_i (cf. (5.4)); the latter is possible as the sets $L_{\mathrm{PREF}}, L_{\mathrm{MID}}$ and L_{SUFF} are PSPACE-enumerable.

6. PSPACE LOWER BOUND

Recall that a language is *definite* if it is a finite Boolean combination of languages of the form $w\Sigma^*$, for $w \in \Sigma^*$. In this section we prove the following result which, in particular, implies the lower bound of Theorem 1:

Theorem 27. *For every class $\mathcal{F}$ containing all definite languages, the $\mathcal{F}$ separability problem for languages of OCN is PSPACE-hard.*

A convenient PSPACE-hard problem, to be reduced to $\mathcal{F}$ separability of OCN, can be extracted from [15]. Given an OCA $\mathcal{A}$ and $b \in \mathbb{N}$ presented in binary, the *bounded non-emptiness problem* asks whether $\mathcal{A}$ accepts some word by a *b-bounded* run; a run is b-bounded if counter values along the run are at most b.

Theorem 28 [15]. *The bounded non-emptiness problem is PSPACE-complete, for $\mathcal{A}$ and b represented in binary.*

A detailed analysis of the proof reveals that the problem remains PSPACE-hard even if the input OCA $\mathcal{A} = (Q, \alpha_0, \alpha_f, T, T_{=0})$ is assumed to be *acyclic*, in the sense that there is no reachable configuration α with a non-empty run $\alpha \longrightarrow \alpha$. Observe that an acyclic OCA has no b-bounded run longer than $b|Q|$, a property which will be crucial for the correctness of our reduction.

Proposition 29. *The bounded non-emptiness problem is PSPACE-complete, for acyclic $\mathcal{A}$ and b represented in binary.*

We are now ready to prove Theorem 27, by reduction from bounded non-emptiness of acyclic OCA. Given an acyclic OCA $\mathcal{A} = (Q, (q_0, 0), (q_f, 0), T, T_{=0})$ and $b \in \mathbb{N}$, we construct in polynomial time two OCN $\mathcal{B}$ and $\mathcal{B}'$, with the following properties:

(a) if $\mathcal{A}$ has a b-bounded accepting run then $L(\mathcal{B}) \cap L(\mathcal{B}') \neq \varnothing$ (and thus $L(\mathcal{B})$ and $L(\mathcal{B}')$ are not $\mathcal{F}$ separable);

(b) if $\mathcal{A}$ has no b-bounded accepting run then $L(\mathcal{B})$ and $L(\mathcal{B}')$ are $\mathcal{F}$ separable.

The two OCN $\mathcal{B}$ and $\mathcal{B}'$ will jointly simulate a b-bounded run of $\mathcal{A}$, obeying an invariant that the counter value v of $\mathcal{B}$ is the same as the counter value of $\mathcal{A}$, while the counter value of $\mathcal{B}'$ is $b - v$. The actual input alphabet of $\mathcal{A}$ is irrelevant; as the input alphabet of $\mathcal{B}$ and $\mathcal{B}'$ we take $\Sigma = T \cup T_{=0}$. The OCN $\mathcal{B}$ behaves essentially as $\mathcal{A}$, except that it always allows for a zero test. Formally, $\mathcal{B} = (Q, (q_0, 0), (q_f, 0), U)$, where the transitions U are defined as follows. For every transition $t = (q, a, q', z) \in T$, there is a corresponding transition $(q, t, q', z) \in U$. Moreover, for every zero test $t = (q, a, q') \in T_{=0}$, there is a transition $(q, t, q', 0) \in U$. On the other hand, the OCN $\mathcal{B}'$ starts in the configuration (q_0, b), ends in (q_f, b), and simulates the

transitions of $\mathcal{A}$ but with the opposite effect. Formally, $\mathcal{B}' = (Q \cup X, (q_0, b), (q_f, b), U')$, for a set X of auxiliary states. For every transition $t = (q, a, q', z) \in T$, there is a corresponding transition $(q, t, q', -z) \in U$ with the effect $-z$ opposite to the effect of t. Moreover, for every zero test $t = (q, a, q') \in T_{=0}$, we include into U' the following three transitions

$$(q, \varepsilon, p, -b) \qquad (p, \varepsilon, p', +b) \qquad (p', t, q', 0),$$

for some auxiliary states p, p'. The aim of the first two transitions is to allow the last one only if the counter value is at least b (and thus exactly b, assuming there is also a run of $\mathcal{B}$ on the same input).

We need to argue that the implications (a) and (b) hold. The first one is immediate: every b-bounded accepting run of $\mathcal{A}$ is faithfully simulated by $\mathcal{B}$ and $\mathcal{B}'$, and thus the languages $L(\mathcal{B})$ and $L(\mathcal{B}')$ have non-empty intersection.

For the implication (b), suppose $\mathcal{A}$ has no b-bounded accepting run. The first step is to notice that the languages $L(\mathcal{B})$ and $L(\mathcal{B}')$ are necessarily disjoint. Indeed, any word $w \in L(\mathcal{B}) \cap L(\mathcal{B}')$ would describe a b-bounded accepting run of $\mathcal{A}$: $\mathcal{B}$ ensures that the counter remains non-negative, while $\mathcal{B}'$ ensures that the counter does not increase beyond b and that the zero tests are performed correctly.

Let L contain all prefixes of words from $L(\mathcal{B})$, and likewise L' for $L(\mathcal{B}')$. Let $n = b|Q|$. Recall that due to acyclicity, $\mathcal{A}$ has no b-bounded run of length n (in the sense of the number of transitions) or longer. Thus, for the same reason as above, the intersection $L \cap L'$ contains no word of length n or longer.

In simple words, we are going to show that for every word $w \in L(\mathcal{B}) \cup L(\mathcal{B}')$, looking at the prefix of length n of w suffices to decide whether $w \in L(\mathcal{B})$ or $w \in L(\mathcal{B}')$.

We define a language $K \in \mathcal{F}$ as follows:

$$K \ := \ \left(L(\mathcal{B}) \cap \Sigma^{<n} \right) \ \cup \ \bigcup_{w \in L, |w|=n} w\Sigma^*,$$

where $\Sigma^{<n}$ stands for the set of all words over Σ of length strictly smaller than n, and $|w|$ denotes the length of w. The language K belongs to $\mathcal{F}$ indeed, as $\mathcal{F}$ is closed under finite unions, and every singleton $\{w\}$ belongs to $\mathcal{F}$, due to

$$\{w\} = w\Sigma^* - \bigcup_{a \in \Sigma} wa\Sigma^*.$$

It remains to argue that K separates $L(\mathcal{B})$ and $L(\mathcal{B}')$. By the very definition $L(\mathcal{B}) \subseteq K$, as K contains all words from $L(\mathcal{B})$ of length strictly smaller than n, and all words starting with a prefix, of length n, of a word from $L(\mathcal{B})$. For disjointness of K and $L(\mathcal{B}')$, observe that the languages $L(\mathcal{B}) \cap \Sigma^{<n}$ and $L(\mathcal{B}')$ are disjoint, as already $L(\mathcal{B})$ and $L(\mathcal{B}')$ are. Moreover, for every $w \in L$ of length $|w| = n$, the languages $w\Sigma^*$ and $L(\mathcal{B}')$ are disjoint, as already the intersection $L \cap L'$ contains no word of length n or longer.

Remark 30. The OCN $\mathcal{B}$ and $\mathcal{B}'$ used in the reduction can be easily made deterministic. On the other hand, by a general result of [7] we learn that regular separability of nondeterministic OCN polynomially reduces to regular separability of *deterministic* OCN, making the latter PSpace-complete too.

7. Undecidability for one counter automata

In this section we prove Theorem 2. The argument is similar to the proof of the previous section, except that instead of reducing a fixed undecidable problem, we provide a polynomial reduction from *every* decidable one. This idea derives from the insight of [19].

A universal model of computation that will be convenient for us is 2-counter machines. A *deterministic 2-counter machine* $\mathcal{M}$ consists of a finite set of *states* Q with distinguished *initial* state $q_0 \in Q$, *accepting* state $q_{\mathrm{acc}} \in Q$ and *rejecting* state $q_{\mathrm{rej}} \in Q$, two *counters* c_1, c_2, and a set of transitions, one per state $q \in Q - \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}$. Thus the accepting state and the rejecting one have no outgoing transitions. There are two types of transitions. Type 1 transitions increment one of the counters ($i \in \{1, 2\}$) by one, and type 2 transitions conditionally decrement one of the counters by one:

> **(type 1) in state q, increment c_i and go to state q';**
> **(type 2) in state q, if $c_i > 0$ then decrement c_i and go to state q', else go to state q''.**

A configuration (q, n_1, n_2) of $\mathcal{M}$ consists of a state q and values $n_1, n_2 \geqslant 0$ of the counters. We write $(q, n_1, n_2) \longrightarrow (q', n_1', n_2')$ if a sequence of transitions leads from configuration (q, n_1, n_2) to (q', n_1', n_2'). We say that $\mathcal{M}$ *accepts* a number $k \in \mathbb{N}$ if $(q_0, k, 0) \longrightarrow (q_{\mathrm{acc}}, 0, 0)$, and *rejects* k if $(q_0, k, 0) \longrightarrow (q_{\mathrm{rej}}, 0, 0)$. Note our specific requirement that acceptance or rejection only happens with both counter values equal to 0. The machine $\mathcal{M}$ is *total* if every $k \in \mathbb{N}$ is either accepted or rejected by $\mathcal{M}$. The language $L(\mathcal{M})$ recognized by $\mathcal{M}$ is set of all numbers accepted by M.

Every decidable language, say over the alphabet $\{0, 1\}$, is recognized by some total, deterministic 2-counter machine, under a suitable encoding. Indeed, every word $w \in \{0, 1\}^*$ can be encoded, using binary representation, as a natural number $n(w)$. It is quite standard to show that then for every total deterministic Turing machine $\mathcal{T}$, there is a total deterministic 2-counter machine $\mathcal{M}$ such that $w \in L(\mathcal{T})$ if, and only if, $2^{n(w)} \in L(\mathcal{M})$.[2] Thus, modulo the encoding, decidable languages are a subclass of (in fact, the same class as) subsets $L \subseteq \mathbb{N}$ of natural numbers recognized by total deterministic 2-counter machines. These subsets $L \subseteq \mathbb{N}$ we call below *decidable problems*.

Let $\mathcal{F}$ be a class of languages containing all definite languages. We are going to show a polynomial time reduction from any decidable problem $L \subseteq \mathbb{N}$ to $\mathcal{F}$ separability of OCA languages. This implies undecidability of the latter problem. Indeed, suppose $\mathcal{F}$ separability of OCA languages is decidable, and let $f : \mathbb{N} \to \mathbb{N}$ be any space constructible function such that $\mathcal{F}$ separability of OCA is decidable in $f(n)$ space, where n is the size of input. As a consequence, every decidable problem $L \subseteq \mathbb{N}$ would be actually decidable in space $f(p(n))$ for some polynomial p. This would contradict the space hierarchy theorem (see, e.g., Thm. 9.3 in [31]), according to which there are problems decidable in space $\mathcal{O}(f(2^n))$ but not in space $o(f(2^n))$.

Proposition 31. *Every decidable problem $L \subseteq \mathbb{N}$ reduces polynomially to the $\mathcal{F}$ separability problem of OCA languages.*

Proof. Let $\mathcal{M}$ be a fixed total deterministic 2-counter machine recognizing a language L. Given $k \in \mathbb{N}$, we construct two OCA $\mathcal{A}_1, \mathcal{A}_2$ with the following properties:
(a) if $k \in L(\mathcal{M})$ then $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \neq \varnothing$ (and thus $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are not $\mathcal{F}$ separable);

---

[2] The exponent arises from the standard simulation of a Turing machine by a 3-counter machine; the latter is further simulated by a 2-counter machine which stores the values of the 3 counters c, d, e in the form $2^c 3^d 5^e$.

(b) if $k \notin L(\mathcal{M})$ then $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are $\mathcal{F}$ separable.

As the input alphabet Σ of $\mathcal{A}_1$ and $\mathcal{A}_2$ we take the set of transitions of $\mathcal{M}$. We define two OCA:

$$\mathcal{A}_1 = (Q, (q_0, k), (q_{\mathrm{acc}}, 0), T_1, T_{1,=0}),$$
$$\mathcal{A}_2 = (Q, (q_0, 0), (q_{\mathrm{acc}}, 0), T_2, T_{2,=0}),$$

where transitions T_1 (resp. T_2) and zero tests $T_{1,=0}$ (resp. $T_{2,=0}$) are, roughly speaking, transitions of $\mathcal{M}$ where the second (resp. first) counter is ignored. Formally, for every transition t of M of type 1 on counter c_1 of the form

(type 1) in state q, increment c_1 and go to state q',

there is a transition $(q, t, q', +1) \in T_1$; and for every transition t of M of type 1 on counter c_2 of the form

(type 1) in state q, increment c_2 and go to state q',

there is a transition $(q, t, q', 0) \in T_1$. Furthermore, for every transition t of M of type 2 on counter c_1 of the form

(type 2) in state q, if $c_1 > 0$ then decrement c_1 and go to state q', else go to state q'',

we include the following transition and zero test:

$$(q, t, q', -1) \in T_1 \qquad (q, t, q'') \in T_{1,=0}.$$

Finally, for every transition t of M of type 2 on counter c_2 of the form

(type 2) in state q, if $c_2 > 0$ then decrement c_2 and go to state q', else go to state q'',

we include the following two transitions:

$$(q, t, q', 0) \in T_1 \qquad (q, t, q'', 0) \in T_1.$$

Transitions and zero tests of $\mathcal{A}_2$ are defined symmetrically, with the roles of c_1 and c_2 swapped.

We need to argue that the implications (a) and (b) hold. The first one is immediate: every sequence of transitions of $\mathcal{M}$ leading from $(q_0, k, 0)$ to $(q_{\mathrm{acc}}, 0, 0)$, treated as a word over Σ, belongs both to $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$.

In order to prove implication (b), suppose $k \notin L(\mathcal{M})$. We first observe that $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are necessarily disjoint; indeed, any $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ is a sequence of transitions that accepts k.

As $\mathcal{M}$ is total by assumption, we know that $(q_0, k, 0) \longrightarrow (q_{\mathrm{rej}}, 0, 0)$ in $\mathcal{M}$; let n be the length of the corresponding sequence of transitions.

Let L_1 contain all prefixes of words from $L(\mathcal{A}_1)$, and likewise L_2 for $L(\mathcal{A}_2)$. It is crucial to observe that the intersection $L_1 \cap L_2$ contains no word of length n or longer. Indeed, any $w \in L_1 \cap L_2$ is a sequence of transitions of $\mathcal{M}$ starting from $(q_0, k, 0)$, and thus cannot be longer than n. Moreover $w \in L_1 \cap L_2$ cannot lead, as a sequence of transitions of $\mathcal{M}$, to the rejecting state (as it has no outgoing transitions), and thus w can not have length n either.

The rest of the proof is along the same lines as in the previous section. In simple words, we claim that for a word of length n or longer, it is enough to inspect its prefix of length n in order to classify the word between $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$. Formally, we define a language $K \in \mathcal{F}$ as follows:

$$K := \left(L(\mathcal{A}_1) \cap \Sigma^{<n} \right) \cup \bigcup_{w \in L_1, |w|=n} w\Sigma^*.$$

The language K belongs to $\mathcal{F}$ for the reasons discussed in the previous section. It remains to argue that K separates $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$. By the very definition $L(\mathcal{A}_1) \subseteq K$, as K contains all words from $L(\mathcal{A}_1)$ of length strictly smaller than n, and all words starting with a prefix, of length n, of a word from $L(\mathcal{A}_1)$. For disjointness of K and $L(\mathcal{A}_2)$, observe that the languages $L(\mathcal{A}_1) \cap \Sigma^{<n}$ and $L(A_2)$ are disjoint, as already $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are. Moreover, for every $w \in L_1$ of length $|w| = n$, the languages $w\Sigma^*$ and $L(\mathcal{A}_2)$ are disjoint, as already the intersection $L_1 \cap L_2$ contains no word of length n or longer. □

Remark 32. Theorem 2 is used in [7] to prove undecidability of the regular separability problem for visibly one counter automata (cf. Theorem 5 in [7]). The proof assumes that the alphabets of the input visibly one counter automata can be different; on the other hand, when two input visibly one counter automata are assumed to be over the same alphabet (i.e., they perform increment and decrement operations on the same input letters) the problem seems to be decidable [24]. This shows that the decidability border is quite subtle. In addition, the regular separability problem becomes once more undecidable when one extends visibly one-counter automata over the same alphabet to visibly pushdown automata over the same alphabet, as shown in [22].

8. Final remarks

Our main contribution is to show that the regular separability problem for OCN is decidable (we also provide tight complexity estimation of the problem, namely PSpace-completeness, which we consider however less significant), but it becomes undecidable for OCA (when zero tests are allowed). We believe that this reveals a delicate decidability borderline. For instance recall (cf. Remark 11) that the concept of n-approximation, a core technical ingredient of our decidability proof, still works for OCA, including the Approximation Lemma, but is not prone to effective testing. Below we discuss in more detail two other aspects: relation to the regularity problem for OCN, and obstacles towards extending our approach to regular separability of the many-dimensional extension of OCN, i.e., of VASS.

Undecidability of regularity. Our decidability result contrasts with undecidability of the regularity problem for OCN (given an OCN $\mathcal{A}$, decide if $L(\mathcal{A})$ is regular?), shown in [33]. The proof of [33] works for OCN accepting by final configuration (as assumed in this paper, cf. Section 2), but not for OCN accepting solely by final state. But even in this weaker model the regularity problem is undecidable, as discovered recently by James Worrell [34]. The proof is by reduction from finiteness of the reachability set of a lossy counter machine, which is an undecidable problem [30]. Consider a standard encoding of runs of such a machine as words, and consider the language of *reverses* of such encodings, i.e., encodings read backward. It is not difficult to prove that the language is regular if, and only if, the reachability set of the lossy counter machine is finite. Moreover, one can construct an OCN that recognizes the complement of the language.

Towards regular separability of VASS. Our decidability proof builds upon a notion of n-approximation: an OCN $\mathcal{A}$ is over-approximated by an NFA $\mathcal{A}_n$ which remembers the counter value of $\mathcal{A}$ exactly only below n, and modulo n above this threshold. Could one define n-approximation $\mathcal{V}_n$ of a VASS $\mathcal{V}$ by treating all the counters of $\mathcal{V}$ in that way? In particular, such n-approximation would commute with the synchronized product: $\mathcal{V}_n \otimes \mathcal{U}_n = (\mathcal{V} \otimes \mathcal{U})_n$ for two VASS $\mathcal{V}$ and $\mathcal{U}$ (we extend here naturally the synchronized product operation).

The Approximation Lemma (cf. Lemma 8), quite surprisingly, does not hold for so defined notion of over-approximation. Indeed, the Approximation Lemma would imply that regular separability of $\mathcal{V}$ and $\mathcal{U}$ is equivalent to disjointness of languages of $\mathcal{V}_n$ and $\mathcal{U}_n$, for some $n > 0$ (cf. Corollary 10), which is the same as $L(\mathcal{V}_n \otimes \mathcal{U}_n) = L((\mathcal{V} \otimes \mathcal{U})_n) = \varnothing$ for some $n > 0$; and finally, the latter condition would be equivalent, again due to the Approximation Lemma, to $L(\mathcal{V} \otimes \mathcal{U}) = \varnothing$, which is the same as the languages of $\mathcal{V}$ and $\mathcal{U}$ being disjoint. Thus regular separability of $\mathcal{V}$ and $\mathcal{U}$ would be equivalent to disjointness of $\mathcal{V}$ and $\mathcal{U}$, which is not true in general.

The decidability status of the regular separability problem for VASS languages remains thus open.

OCN versus OCA. As remarked in Section 3, the Approximation Lemma and its consequences (Corollaries 9 and 10) can be shown for OCA just as well. This observation seems to open the way to decidability of the regular separability problem when one of input devices is OCA and the other one is OCN. We conjecture the problem to be decidable, and even to belong to PSpace, relying on the effective semi-linearity of the reachability relation for OCN with one zero test [16].

Acknowledgment

References

[1] J. Almeida. Some algorithmic problems for pseudovarieties. *Publ. Math. Debrecen*, 54:531–552, 1999.

[2] Mohamed Faouzi Atig, Dmitry Chistikov, Piotr Hofman, K. Narayan Kumar, Prakash Saivasan, and Georg Zetzsche. The complexity of regular abstractions of one-counter languages. In *Proc. LICS'16*, pages 207–216, 2016.

[3] Mohamed Faouzi Atig, Dmitry Chistikov, Piotr Hofman, K. Narayan Kumar, Prakash Saivasan, and Georg Zetzsche. Complexity of regular abstractions of one-counter languages. *CoRR*, abs/1602.03419, 2016.

[4] Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *Proc. LICS'15*, pages 32–43, 2015.

[5] Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. On the expressiveness of Parikh automata and related models. In *Proc. NCMA'11*, pages 103–119, 2011.

[6] Dmitry Chistikov and Christoph Haase. The taming of the semi-linear set. In *Proc. ICALP'16*, pages 128:1–128:13, 2016.

[7] Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Regular separability of Parikh automata. In *Proc. ICALP'17*, pages 117:1–117:13, 2017.

[8] Lorenzo Clemente, Wojciech Czerwiński, Sławomir Lasota, and Charles Paperman. Separability of reachability sets of vector addition systems. In *Proc. STACS'17*, pages 24:1–24:14, 2017.

[9] Wojciech Czerwinski and Slawomir Lasota. Regular separability of one counter automata. In *Proc. LICS'17*, pages 1–12, 2017.

---

[3] Found out at *Autobóz'16*, the annual research camp of Warsaw automata group and friends.

[10] Wojciech Czerwinski, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular separability of well-structured transition systems. In *Proc. CONCUR'18*, pages 35:1–35:18, 2018.

[11] Wojciech Czerwiński, Wim Martens, and Tomás Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Proc. ICALP'13*, pages 150–161, 2013.

[12] Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, and Marc Zeitoun. A note on decidable separability by piecewise testable languages. In *Proc. FCT'15*, pages 173–185, 2015.

[13] Eric Domenjoud. Solving systems of linear Diophantine equations: An algebraic approach. In *Proc. MFCS'91*, pages 141–150, 1991.

[14] Emanuele D'Osualdo, Roland Meyer, and Georg Zetzsche. First-order logic with reachability for infinite-state systems. In *Proc. LICS'16*, pages 457–466, 2016.

[15] John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015.

[16] Alain Finkel, Jérôme Leroux, and Grégoire Sutre. Reachability for two-counter machines with one test and one reset. In *Proc. FSTTCS'18*, pages 31:1–31:14, 2018.

[17] Jean Goubault-Larrecq and Sylvain Schmitz. Deciding piecewise testable separability for regular tree languages. In *Proc. ICALP'16*, pages 97:1–97:15, 2016.

[18] John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979.

[19] Harry B. Hunt III. On the decidability of grammar problems. *Journal of the ACM*, 29(2):429–447, 1982.

[20] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, 1969.

[21] Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In *Proc. ICALP'03*, pages 681–696, 2003.

[22] Eryk Kopczyński. Invisible pushdown languages. In *Proc. LICS'16*, pages 867–872, 2016.

[23] Eryk Kopczynski and Anthony Widjaja To. Parikh images of grammars: Complexity and applications. In *Proc. LICS'10*, pages 80–89, 2010.

[24] Christof Löding. Personal communication, 2017.

[25] Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. In *Proc. FSTTCS'13*, pages 363–375, 2013.

[26] Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In *Proc. MFCS'13*, pages 729–740, 2013.

[27] Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In *Proc. ICALP'14*, pages 342–353, 2014.

[28] Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Logical Methods in Computer Science*, 12(1), 2016.

[29] Loic Pottier. Minimal solutions of linear Diophantine systems: Bounds and algorithms. In *Proc. RTA'91*, pages 162–173, 1991.

[30] Philippe Schnoebelen. Lossy counter machines decidability cheat sheet. In *Proc. RP'10*, pages 51–75, 2010.

[31] Michael Sipser. *Introduction to the theory of computation*. Cengage Learning, 2013.

[32] Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2):231–250, 1976.

[33] Rüdiger Valk and Guy Vidal-Naquet. Petri nets and regular languages. *J. Comput. Syst. Sci.*, 23(3):299–325, 1981.

[34] James Worrell. Personal communication, 2016.