A Note on Decidable Separability by Piecewise Testable Languages

Wojciech Czerwiński [1] and Wim Martens [2]

[1] University of Warsaw
[2] Institute for Computer Science, University of Bayreuth
wczerwin@mimuw.edu.pl, wim.martens@uni-bayreuth.de

Abstract. The separability problem for languages from a class $\mathcal{C}$ by languages of a class $\mathcal{S}$ asks whether, for two given word languages I and E from $\mathcal{C}$, there exists a language S from $\mathcal{S}$ which includes I and excludes E, that is, $I \subseteq S$ and $S \cap E = \emptyset$. It is known that separability for context-free languages by any class containing all definite languages (such as regular languages) is undecidable. We show that separability of context-free languages by piecewise languages is decidable. This may be remarkable in the light that it is undecidable whether a context-free language is piecewise testable. We generalize this decidability result by showing that, for every class of languages that is robust under some rather weak operations and has decidable emptiness and diagonal problems, separability with respect to piecewise testable languages is decidable. An example of such a class is the class of languages defined by labeled vector addition systems.

1 Introduction

We say that language I can be *separated* from E by a language S if S includes I and excludes E, that is, $I \subseteq S$ and $S \cap E = \emptyset$. In this case, we call S is a *separator*. We study the *separability problem* of classes $\mathcal{C}$ by classes $\mathcal{S}$:

Given: Two languages I and E from a class $\mathcal{C}$.
Question: Can I and E be separated by some language from $\mathcal{S}$?

Separability is a classical problem in mathematics and computer science that recently found much new interest. For example, recent work investigated the separability problem of regular languages by piecewise testable languages [9,26], locally testable and locally threshold testable languages [25] or by first order definable languages [28]. Another recent example which uses separation and goes beyond regularity, is the proof of Leroux [16] for the decidability of reachability for vector addition systems or petri nets. This proof simplifies earlier proofs by Mayr [19] and Kosaraju [15].

In this paper we focus on the theoretical underpinnings of separating languages by piecewise testable languages. Our interest in piecewise testable languages is mainly because of the following two reasons. First, it was shown recently

[9,26] that separability of regular languages (given by their non-deterministic automata) by piecewise testable languages is in PTIME. We found the tractability of this problem to be rather surprising.

Second, piecewise testable languages are a very natural class of languages in the sense that they only reason about the *order* of symbols. More precisely, they are finite Boolean combinations of regular languages of the form $A^*a_1A^*a_2A^*\cdots A^*a_nA^*$ in which $a_i \in A$ for every $i = 1, \ldots, n$ [30]. We currently investigate to which extent piecewise testable languages and fragments thereof can be used for computing simple explanations for the behavior of possibly complex systems. We feel that such languages may have potential to be understandable for humans who, e.g., debug a system.[3]

Separation and Characterization For classes $\mathcal{C}$ effectively closed under complement, separation of $\mathcal{C}$ by $\mathcal{S}$ is a natural generalization of the *characterization problem* for $\mathcal{C}$ by $\mathcal{S}$, which is defined as follows. For a given language $L \in \mathcal{C}$ decide whether L is in $\mathcal{S}$. Indeed, L is in $\mathcal{S}$ if and only if L can be separated from its complement by a language from $\mathcal{S}$.

The characterization problem has been heavily investigated. The starting points were famous works of Schützenberger [29] and Simon [30], which have solved the characterization problem for the regular languages and the first-order definable languages and piecewise testable languages, respectively. There were many more results showing that, for regular languages and some its subclass, often of languages definable in a given logic, the problem is decidable, see for example [3, 22, 24, 27, 31, 33]. Similar problems have been considered for trees [1, 5, 6].

Decidability To the best of our knowledge, all the above work and in general all the decidable characterizations were obtained in cases where the class $\mathcal{C}$ is regular, or a subclass of regular languages.

This could be due to several negative results which seem to form an unbreakable barrier for any investigation beyond regular languages. For a context-free language (given by a grammar or a pushdown automaton) it is undecidable to determine whether it is a regular language by Greibach's theorem [12]. Furthermore, it is also undecidable to determine whether a given context-free language is piecewise testable. This is not an immediate consequence of Greibach's theorem as far as we know but can be proved similarly.[4]

Regarding separability for context-free languages, there is a strong connection between the intersection emptiness problem and separability. Trivially, testing intersection emptiness of two given context-free languages is the same as deciding if they can be separated by some context-free language. However, in general, the negative result is even more overwhelming. Hunt [14] proved that separability of context-free languages by any class containing all the *definite languages* is

[3] We provide more concrete scenarios in the appendix for interested reviewers.
[4] We provide proof in the Appendix.

undecidable. A language L is *definite* if it can be defined as $L = F_1A^* \cup F_2$, where F_1 and F_2 are finite languages over alphabet A. As such, for definite languages, it can be decided whether a given word w belongs to L by looking at the prefix of w of a given fixed length. Of course the analogous statement can be done for *reverse definite* languages, in which we are looking at suffixes. Containing all the definite, or reverse definite, languages is a very weak condition. Note that if a logic can test what is the i-th letter of a word and it admits a boolean combination it already can define all the definite languages. In his paper, Hunt makes an explicit link between intersection emptiness and separability. Hunt says:

> "We show that separability is undecidable in general for the same reason that the emptiness-of-intersection problem is undecidable. Therefore, it is unlikely that separability can be used to circumvent the undecidability of the emptiness-of-intersection problem."

Our Contribution In this paper we essentially show that the above mentioned quote does not apply for separability by piecewise testable languages. In particular, we will show that it can be decided whether two given context-free languages are separable by a piecewise testable language. This may come as a surprise in the light of the undecidability results we already discussed.

Actually, we prove a stronger result that implies that separability by piecewise testable languages is also decidable for some rather expressive classes such as Petri net languages (or, alternatively, labeled vector addition system languages). Basically, this result is a reduction from separability by piecewise testable languages to a problem that we call *diagonal problem*.

The proof is rather simple but we feel that its implications are surprising. It shows that deciding separability by a non-trivial class such as piecewise testable languages is possible beyond regular languages. Alternatively it can be seen as an evidence that piecewise testable languages are in some way extraordinary and enjoy nice properties when it comes to computational complexity.

A curiosity of our work is perhaps the absence of algebraic methods. Most of the previous decidability results have considered syntactic monoids of regular languages and investigated properties thereof. Some exceptions are the recent studies of separability of regular languages by piecewise testable languages [9,26]. However, since the algebraic framework for regular languages is so rich, some may not find it clear at first sight whether or not these recent works do or do not rely on algebraic methods; perhaps simply in a rephrased way.

Here, the situation is different, in the sense that for context-free languages the syntactic monoid does not exist and it is hard to design any algebraic framework for them. So the work shows that it is not always necessary to use algebraic techniques to prove separability questions. Of course, we do not want to express our favor of one method above another. The richness and depth of the algebraic framework are without doubt.

Structure of the paper In Section 2 we introduce basic notions and notation, the *diagonal problem*, and state our main lemma. Section 3 is devoted to an algorithm for separability, which is essentially a reduction of separability to the diagonal problem. In Section 4 we discuss applications of the lemma, in particular we show that it applies to context-free languages and languages of labeled vector addition systems (alternatively, languages of labeled Petri nets).

2 Preliminaries

The set of all integers is denoted by $\mathbb{Z}$, the set of all nonnegative integers is denoted by $\mathbb{N}$. A *word* is a concatenation $w = a_1 \cdots a_n$ of symbols a_i that come from a finite alphabet A. The *length* of word w is n, the number of its letters. The *alphabet* of w is the set of all letters $\{a_1, \ldots, a_n\}$ occurring in w and is denoted by $\mathrm{Alph}(w)$. For a subalphabet $B \subseteq A$, a word $v \in A^*$ is a *B-subsequence* of w, denoted $v \preceq_B w$, if $v = b_1 \cdots b_m$ and $w \in B^* b_1 B^* \cdots B^* b_m B^*$. Notice that we do not require that $\{b_1, \ldots, b_m\} \subseteq B$ or $B \subseteq \{b_1, \ldots, b_m\}$. We refer to the relation $\preceq_A$ simply as the *subsequence* relation and denote it by $\preceq$.

A regular language over alphabet A is a *piece language* if it is of the form $A^* a_1 A^* \cdots A^* a_n A^*$ for some $a_1, \ldots, a_n \in A$, that is, it is the set of words which have $a_1 \cdots a_n$ as a subsequence. A regular language is a *piecewise testable language* if it is a (finite) boolean combination of piece languages. The class of all piecewise testable languages is denoted PTL.

A Sufficient Condition for Decidable Separability In Section 3 we will present an algorithm that can decide separability for language classes that are robust under some operations and have a decidable emptiness and diagonal problem. We explain these notions next.

Fix a language L over alphabet A. For an alphabet B, a *B-projection* of a word is its (longest) subsequence consisting of letters from B. The *B-projection* of a language L is the set of all B-projections of words belonging to L. Therefore, the B-projection of L is a language over alphabet $A \cap B$. The *B-upward closure* of a language L is the set of all words which have a B-subsequence in L, i.e.,

$$\{w \in (A \cup B)^* \mid \exists v \in L \text{ such that } v \preceq_B w\}.$$

In other words, the B-upward closure of L consists of all words that can be obtained from taking a word in L and padding it with symbols from B.

A class of languages $\mathcal{C}$ is *closed* under an operation OP if $L \in \mathcal{C}$ implies that $\mathrm{OP}(L) \in \mathcal{C}$. We use term *effectively closed* if the representation of $\mathrm{OP}(L)$ can be effectively computed from the representation of L.

A class $\mathcal{C}$ of languages is *robust* if it is effectively closed under:

1. B-projection for every finite alphabet B,
2. B-upward closure for every finite alphabet B, and
3. intersection with regular languages.

The lemma we will prove requires two decidability properties. One is decidable emptiness. The *emptiness problem* for a language L simply asks whether $L = \emptyset$. The other problem we require to be decidable is the *diagonal problem*, which we explain next. Let $A = \{a_1, \ldots, a_n\}$. For a letter $a \in A$ and a word $w \in A^*$, let $\#_a(w)$ denote the number of occurrences of a in w. The *Parikh image* of a word w is the n-tuple

$$(\#_{a_1}(w), \ldots, \#_{a_n}(w)).$$

The *Parikh image* of a language L is the set of all Parikh images of words from L. A tuple $(m_1, \ldots, m_n) \in \mathbb{N}^n$ is *dominated* by a tuple $(d_1, \ldots, d_n) \in \mathbb{N}^n$ if $d_i \geq m_i$ for every $i = 1, \ldots, n$. The *diagonal problem* for language L asks whether there exist infinitely many $m \in \mathbb{N}$ such that the tuple $(m, \ldots, m)$ is dominated by some tuple in the Parikh image of L. We are now ready to state our main lemma:

Lemma 1. *Let $\mathcal{C}$ be a robust class of languages with decidable emptiness problem. If the diagonal problem is decidable for $\mathcal{C}$, then separability of $\mathcal{C}$ by PTL is decidable.*

We prove the lemma in Section 3. In fact, Section 3 simply presents an algorithm to decide separability and uses the conditions as stated in the lemma.

3 The Algorithm for Separability

In this section we prove Lemma 1. Fix two languages I and E from a class $\mathcal{C}$. By the lemma statement, we may assume that $\mathcal{C}$ is robust and has decidable emptiness and diagonal problems.

In order to test whether I is separable from E they are separable by a piecewise testable language S, we run in parallel two semi-procedures. The *positive* one looks for a witness that I and E are separable by PTL, whereas the *negative* one looks for a witness that they are *not* separable by a PTL. Since one of the semi-procedures always terminates, we have an effective algorithm that decides separability. It remains to describe the two semi-procedures.

Positive semi-procedure The positive semi-procedure simply enumerates all the piecewise testable languages over the union of the alphabets of I and E. For every piecewise testable language S it checks whether S is a separator, so if $I \subseteq S$ and $E \cap S = \emptyset$. The first test is equivalent to $I \cap (A^* \setminus S) = \emptyset$. Thus both tests boil down to checking whether the intersection of a language from the class $\mathcal{C}$ (I or E, respectively) and a regular language (S and $A^* \setminus S$, respectively) is empty. This is decidable, as $\mathcal{C}$ is effectively closed under taking intersections with regular languages and has a decidable emptiness problem.

Negative semi-procedure In order to design the negative semi-procedure we need to use a characterization stating when I and E are not separable by a

PTL. One such characterization that fits our purpose very well can be derived easily from [32]. For stating it we need to define a few notions. For a subalphabet $B \subseteq A$ we denote

$$B^{\otimes} = \{w \mid \mathrm{Alph}(w) = B\},$$

i.e., $B^{\otimes}$ is the set of words which contain all and only the letters from B. A *pattern* $(\overrightarrow{u}, \overrightarrow{B})$ consists of

$$\overrightarrow{u} = (u_0, \ldots, u_k) \quad \text{and} \quad \overrightarrow{B} = (B_1, \ldots, B_k),$$

where $u_i \in A^*$ and $B_i \subseteq A$ for $i \in \{1, \ldots, k\}$. For a pattern $(\overrightarrow{u}, \overrightarrow{B})$ and a number $n \in \mathbb{N}$ we define a language

$$\mathcal{L}(\overrightarrow{u}, \overrightarrow{B}, n) = u_0 \ (B_1^{\otimes})^n \ u_1 \cdots u_{k-1} \ (B_k^{\otimes})^n \ u_k.$$

We are now ready to formulate the characterization.

Lemma 2 (Implicit in van Rooijen and Zeitoun [32]). *Languages I and E are not separable by piecewise testable languages if and only if there exists a pattern $(\overrightarrow{u}, \overrightarrow{B})$ such that for every $n \in \mathbb{N}$ the language $\mathcal{L}(\overrightarrow{u}, \overrightarrow{B}, n)$ has a nonempty intersection with both I and E.*

We remark that this lemma does not assume anything on I and E except for the fact that they are languages. In particular, I and E are not required to be regular. As a minor remark, we note that the proof of this lemma does not rely on algebraic methods.

Lemma 2 is neither explicitly stated nor proved in [32], but Lemma 6 in [32] is very similar. It is formulated for regular languages I and E but, in order to prove Lemma 2 it is enough to literally repeat the proof of Lemma 6 from [32] without its last sentence.

Lemma 2 shows that there is always a finite witness for inseparability: a pattern $(\overrightarrow{u}, \overrightarrow{B})$. Therefore the negative semi-procedure enumerates all the possible patterns and for every one checks the condition from Lemma 2. It remains to show how to decide this condition, i.e., for a given pattern $(\overrightarrow{u}, \overrightarrow{B})$ test whether for all $n \in \mathbb{N}$ the intersection of $\mathcal{L}(\overrightarrow{u}, \overrightarrow{B}, n)$ with both I and E is nonempty.

Checking the condition Here we show for an arbitrary language from $\mathcal{C}$ how to check whether for all $n \in \mathbb{N}$ its intersection with the language $\mathcal{L}(\overrightarrow{u}, \overrightarrow{B}, n)$ is nonempty. Let us fix $L \in \mathcal{C}$ over an alphabet A and a pattern $(\overrightarrow{u}, \overrightarrow{B})$, where $\overrightarrow{u} = (u_0, \ldots, u_k)$ and $\overrightarrow{B} = (B_1, \ldots, B_k)$. Intuitively, the problem we consider is just a diagonal problem with some artifacts: we are counting the number of occurrences of alphabets B_i and checking whether those numbers can simultaneously become arbitrarily big.

We show decidability of the non-separability problem by a formal reduction to the diagonal problem. We perform a sequence of steps. In every step we will slightly modify the considered language L and appropriately customize the

condition to be checked. Using the closure properties of the class $\mathcal{C}$ we will assure that the investigated language still belongs to $\mathcal{C}$.

First we add special letters $\$_i$, for $i \in \{1, \ldots, k\}$, which do not occur in A. These letters are meant to count how many times alphabet B_i is occurring in the word. Then we will assure that words are of the form

$$u_0 \, (B_1 \cup \{\$_1\})^* \, u_1 \cdots u_{k-1} \, (B_k \cup \{\$_k\})^* \, u_k,$$

which already is close to what we need for the pattern. Then we will check that in between every two letters $\$_i$ (with the same i), every letter from B_i occurs, so that the $\$_i$ are indeed counting the number of iterations through the entire alphabet B_i. Finally we will remove all the letters except those from $\{\$_1, \ldots, \$_k\}$. The resulting language will contain only words of the form $\$_1^*\$_2^* \cdots \$_k^*$ and the condition to be checked will be exactly the diagonal problem.

More formally, let $L_0 := L$. We consider a sequence of modifications starting from L_0, resulting in L_1, L_2, L_3, and L_4. Each of them will be in $\mathcal{C}$ and we describe them next.

Language L_1 is the $\{\$_1, \ldots, \$_k\}$-upward closure of L_0. The idea is that L_1 contains, in particular, all words where the $\$_i$ are placed "correctly", that is, in between two $\$_i$ letters the whole alphabet B_i should occur. However at this moment we do not check it. By the closure under B-upward closures, language L_1 belongs to $\mathcal{C}$.

Note that L_1 also contains words in which the $\$_i$ letters are placed totally arbitrary. In particular, they can occur in the wrong order. The idea behind L_2 is to consider only those words, in which the $\$_i$ were guessed at least in the good areas. Concretely, L_2 is an intersection of L_1 with the language

$$u_0 \, (B_1 \cup \{\$_1\})^* \, u_1 \cdots u_{k-1} \, (B_k \cup \{\$_k\})^* \, u_k.$$

By the closure under intersection with regular languages L_2 belongs to $\mathcal{C}$.

Language L_2 still may contain words, such that in between two $\$_i$ letters not all the letters from B_i occur, we get rid of these by intersecting L_2 with the regular language

$$u_0(\$_1 B_1^{\otimes})^* \$_1 u_1 \cdots u_{k-1} (\$_k B_k^{\otimes})^* \$_k u_k.$$

As such, we obtain L_3 which, again by closure under intersection with regular languages, belongs to $\mathcal{C}$.[5]

Note that intersection of $L = L_0$ with the language $\mathcal{L}(\overrightarrow{u}, \overrightarrow{B}, n)$ is nonempty if and only if L_3 contains a word with precisely $n+1$ letters $\$_i$ for every $i \in \{1, \ldots, k\}$. Indeed, L_3 just contains the (slightly modified versions of) words from L_0 which fit into the pattern and in which the letters $\$_i$ "count" occurrences of $B_i^{\otimes}$. Furthermore, for every word in L_3, the word obtained by removing some occurrences of some $\$_i$ is in L_3 as well. It is thus enough to focus on the $\$_i$ letters.

[5] Of course, one could also immediately obtain L_3 from L_1 by performing a single intersection with a regular language.

Language L_4 is therefore the $\{\$_1, \ldots, \$_k\}$-projection of L_3. By the closure under B-projections, for every $B \subseteq A$, language L_4 belongs to $\mathcal{C}$.

The words contained in L_4 are therefore of the form

$$\$_1^{a_1} \cdots \$_k^{a_k},$$

such that there exists $w \in L$ with at least $a_i - 1$ occurrences of $B_i^{\otimes}$. Therefore intersection of L with $\mathcal{L}(\overrightarrow{u}, \overrightarrow{B}, n)$ is nonempty for all $n \geq 0$ if and only if the tuple $(n, \ldots, n)$ belongs to the Parikh image of L_4 for infinitely many $n \geq 0$. Since this is precisely the diagonal problem, which is decidable for the class $\mathcal{C}$, the proof of Lemma 1 is complete. □

4 Decidable Classes

In this section we show that separability by piecewise testable languages is decidable for a wide range of classes, by proving that they meet the conditions of Lemma 1. In particular, we show this for context-free languages, languages of labeled vector addition systems (which are the same as languages of labeled Petri nets). We comment also on other natural classes of languages containing all the regular languages.

Theorem 3. *Separability by piecewise testable languages is decidable for*

1. *context-free languages; and for*
2. *languages of labeled vector addition systems.*

We note that our approach also allows to mix the above scenarios. That is, separability of a context-free language from a language of a labeled vector addition system is also decidable. In the remainder of this section, we prove the theorem.

Context-Free Languages It is easy to see that context-free languages are closed under taking B-upward closures and B-projections. Since they are well known to be closed under intersection with regular languages, they are robust. Furthermore, it is well-known that emptiness for context-free languages can be decided in polynomial time.

The only nontrivial condition is deciding the diagonal problem. A set $S \subseteq \mathbb{N}^k$ is *linear* if it is of the form

$$S = \{v + n_1 v_1 + \ldots + n_m v_m \mid n_1, \ldots, n_m \in \mathbb{N}\}$$

for some *base* vector $v \in \mathbb{N}^k$ and *period* vectors $v_1, \ldots, v_m \in \mathbb{N}^k$. A *semilinear* set is a finite union of linear sets. Parikh's theorem [23] states that the Parikh image of a context-free language is semilinear, moreover the computation of its description as a (finite) union of linear sets if effective. It is enough to check whether for infinitely many $n \geq 0$ the mentioned semilinear set contains a tuple

that dominates $(n, \ldots, n)$. It is easy to see that this condition is true for a semilinear set if and only if it is true for some of the linear sets it consists of. A linear set S contains a tuple that dominates $(n, \ldots, n)$ for infinitely many $n \geq 0$ if and only if for every coordinate there exists a period of S which is positive on this coordinate. Indeed, if there is some coordinate such that no period of S is positive on this coordinate then members of S are bounded on this coordinate. On the other hand, if, for every coordinate i, there is a period v_{p_i} which is positive on this coordinate then $v_{p_1} + \cdots + v_{p_k}$ is positive on all coordinates and thus $v + n(v_{p_1} + \cdots + v_{p_k})$ dominates $(n, \ldots, n)$. The above condition is clearly decidable, which means that, by Lemma 1, separability for context-free languages by piecewise testable languages is decidable.

Languages of Labeled Vector Addition Systems and Petri Nets A k-dimensional *labeled vector addition system*, or *labeled VAS* $M = (A, T, \ell, s, t)$ over alphabet A consists of a set of *transitions* $T \subseteq \mathbb{Z}^k$, a labeling $\ell : T \to A \cup \{\varepsilon\}$, where ε stands for the empty word and *source* and *target* vectors $s, t \in \mathbb{N}^k$. A labeled VAS defines a transition relation on the set $\mathbb{N}^k$ of *markings*. For two markings $u, v \in \mathbb{N}^k$ we write $u \xrightarrow{a} v$ if there is $r \in T$ such that $u + r = v$ and $\ell(r) = a$, where the addition of vectors is defined as an addition on every coordinate. For two markings $u, v \in \mathbb{N}^k$ we say that u *reaches* v *via a word* w if there is a sequence of markings $u_0 = u, u_1, \ldots, u_{n-1}, u_n = v$ such that $u_i \xrightarrow{a_i} u_{i+1}$ for all $i \in \{0, \ldots, n-1\}$ and $w = a_0 \cdots a_{n-1}$. For a given labeled VAS M the *language* of M, denoted $L(M)$, is the set of all words $w \in A^*$ such that source reaches target via w. We note that languages of labeled VASs are the same as languages of labeled Petri nets.

Simple closure properties Let $B \subseteq A$. Closure under B-projection is simple. We modify a VAS by changing the label to ε of all the transitions which were previously labeled by a letter outside B. Showing closure under B-upward closure is realized by adding new transitions of the form $(0, \ldots, 0)$ labeled by b, one for each letter $b \in B$.

Closure under intersection with regular languages To show the closure under intersection with regular languages we first observe that every regular language is also a language of a VAS. This follows rather quickly from their definition; see also [11,17]. For the purpose of being self-contained, we provide the argument in the Appendix. Now it suffices to show that languages of VASs are closed under intersection (and therefore also under intersection with regular languages). This can be easily seen by a standard product construction, see the Appendix.

Decidability of emptiness and diagonal problems For a labeled VAS, the emptiness problem of its language is equivalent to the question whether its target is reachable from its source. This problem is called the reachability problem and is decidable for VASs, see [19] or the recent simpler solution [16].

We now argue that the diagonal problem is decidable. First we will show that it is enough to consider VASs in which the target marking equals $(0, \ldots, 0)$.

To this end, let M be a k-dimensional labeled VAS with source vector $s = (s_1, \ldots, s_k)$ and target vector $t = (t_1, \ldots, t_k)$. We transform M to a new VAS M' in which we add two auxiliary coordinates, called *life* coordinates. The source coordinate is enriched by 0 on one life coordinate and by 1 on the other one, so it is $s' = (s_1, \ldots, s_k, 0, 1) \in \mathbb{N}^{k+2}$. Every original transition has two copies. One of these transitions subtracts one from the first life coordinate and adds one to the second life coordinate, the second transition does the opposite. Note that nonemptiness of life coordinates serve just as a necessary condition for firing any transition, as every transition subtracts one from one of these coordinates. Therefore, the original source marking s reaches the original target marking t via the same set of words by which the new source marking s' reaches the either $(t_1, \ldots, t_k, 0, 1)$ or $(t_1, \ldots, t_k, 1, 0)$. We add also two *final* transitions, which subtract the original target vector, subtract one from one of the life coordinates and are labeled by ε. Note that, therefore, s can reach t by a word w in M if and only if s' can reach 0^{k+2} by w in M'. Indeed, the implication from left to right is immediate. On the other hand, in order to reach the marking 0^{k+2} in M', the last transition has to be the final transition, so implication from right to left also holds. Thus it is enough to solve the *zero-reachability problem*, i.e., the reachability problem in which the target marking is of the form $(0, \ldots, 0)$.

We will show that zero-reachability is decidable by a reduction to the place-boundedness problem for VASs with one zero test, which is decidable due to Bonnet et al. [7]. We modify the considered VAS in the following way. For every letter $a \in A$ we add a new *letter-coordinate*, which is counting how many times we read the letter a, that is, for every transition which is labeled by $a \in A$ we write 1 in the letter-coordinate corresponding to a and 0 in the letter-coordinates corresponding to other letters. The set of letter-coordinates computes the Parikh image of a word. We also add one new *minimum-coordinate* and a new transition, which subtracts one from all the letter-coordinates and adds one to the minimum-coordinate. It is easy to see that minimum-coordinate can be maximally reach the minimum number from the Parikh image tuple. Additionally, for every letter-coordinate we add a transition, labeled by ε, which can decrease this coordinate by one. The diagonal problem for the original VAS is equivalent to the question whether for infinitely many $n \geq 0$ the source marking, enriched by zeros in the new coordinates, reaches a marking

$$(0, \ldots, 0, n),$$

with zeros everywhere beside the minimum-coordinate with number n. This can be easily reduced to the place-boundedness for a VAS with one zero test. We do not show the details. Intuitively, the zero test checks whether there are zeros everywhere else than the minimum-coordinate and we check whether under this condition the minimum-coordinate can get unbounded. This finishes the proof of decidability of the diagonal problem for labeled VASs.

Other classes Among another natural language classes extending regular languages one can think about context-sensitive languages, languages of process

rewrite systems (defined by Mayr [21]), which are a common generalization of pushdown automata and VASs. Another interesting class is the class of languages defined by lossy counter machines [20].

Unfortunately context-sensitive languages do not meet the conditions of Lemma 1, as the emptiness problem for this class is undecidable. Indeed, for two given context-free grammars it is easy to describe a context-sensitive grammar generating exactly words which belong to the both context-free grammar. In this way we reduce the undecidable problem asking whether two context-free languages are disjoint to the emptiness problem for context-sensitive grammars.

Decidability for separation of process rewrite systems by piecewise testable languages is left open. Robustness of this class can be shown easily, similarly as in the case of VASs. The emptiness problem is decidable due to [21]. However, we do not know the answer for the decidability of the diagonal problem, which is central to our approach. However, we conjecture that it is decidable.

When we attack the diagonal problem for lossy counter machines in the same way as for VASs, we notice that it seems at least as hard as the finiteness problem, which is unfortunately undecidable. So, we do not know yet if languages of lossy counter machines have a decidable separation with respect to piecewise testable languages.

5 Concluding Remark

Since the decidability results we presented seem to be in strong contrast with the remark of Hunt in the introduction, we briefly comment on this. What we essentially do is show that undecidable emptiness-of-intersection for a class $\mathcal{C}$ does not always imply undecidability for separability of $\mathcal{C}$ with respect to some non-trivial class of languages. In the case of separability with respect to piecewise testable languages, the main reason is basically that we only need to construct intersections of languages from $\mathcal{C}$ with languages that are regular (or even piecewise or co-piecewise testable). Here, the fact that such intersections can be effectively constructed, together with decidable emptiness and diagonal problems seem to be sufficient for decidability.

Acknowledgments We would like to thank Tomáš Masopust for pointing us to [14] and Thomas Place for pointing out to us that determining if a given context-free language is piecewise testable is undecidable.

References

1. Timos Antonopoulos, Dag Hovland, Wim Martens, and Frank Neven. Deciding twig-definability of node selecting tree automata. In *International Conference on Database Theory (ICDT)*, pages 61–73, 2012.
2. Marcelo Arenas, Sebastián Conca, and Jorge Pérez. Counting beyond a yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In *World Wide Web Conference (WWW)*, pages 629–638, 2012.

3. Mustapha Arfi. Polynomial operations on rational languages. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 198–206, 1987.
4. Pablo Barceló. Querying graph databases. In *Symposium on Principles of Database Systems (PODS)*, pages 175–188, 2013.
5. Mikołaj Bojanczyk and Tomasz Idziaszek. Algebra for infinite forests with an application to the temporal logic ef. In *International Conference on Concurrency Theory (CONCUR)*, pages 131–145, 2009.
6. Mikołaj Bojanczyk, Luc Segoufin, and Howard Straubing. Piecewise testable tree languages. *Logical Methods in Computer Science*, 8(3), 2012.
7. Rémi Bonnet, Alain Finkel, Jérôme Leroux, and Marc Zeitoun. Model checking vector addition systems with one zero-test. *Logical Methods in Computer Science*, 8(2), 2012.
8. Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. A graphical query language supporting recursion. In *Proc. of the ACM Special Interest Group on Management of Data (SIGMOD)*, pages 323–330, 1987.
9. Wojciech Czerwinski, Wim Martens, and Tomáš Masopust. Efficient separability of regular languages by subsequences and suffixes. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 150–161, 2013.
10. Facebook. Graph search. https://www.facebook.com/about/graphsearch, 2014.
11. A. Ginzburg and Michael Yoeli. Vector addition systems and regular languages. *J. Comput. Syst. Sci.*, 20(3):277–284, 1980.
12. S. Greibach. A note on undecidable properties of formal languages. *Math. Systems Theory*, 2(1):1–6, 1968.
13. Jelle Hellings. Conjunctive context-free path queries. In *Proc. 17th International Conference on Database Theory (ICDT)*, pages 119–130, 2014.
14. Harry B. Hunt III. On the decidability of grammar problems. *J. ACM*, 29(2):429–447, 1982.
15. S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC)*, pages 267–281, 1982.
16. Jérôme Leroux. The general vector addition system reachability problem by presburger inductive invariants. *Logical Methods in Computer Science*, 6(3), 2010.
17. Jérôme Leroux, Vincent Penelle, and Grégoire Sutre. On the context-freeness problem for vector addition systems. In *ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 43–52, 2013.
18. Katja Losemann and Wim Martens. The complexity of regular expressions and property paths in SPARQL. *ACM Trans. Database Syst.*, 38(4):24, 2013.
19. E. W. Mayr. An algorithm for the general petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.
20. R. Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 297(1-3):337–354, 2003.
21. Richard Mayr. Process rewrite systems. *Electr. Notes Theor. Comput. Sci.*, 7:185–205, 1997.
22. Robert McNaughton. Algebraic decision procedures for local testability. *Mathematical Systems Theory*, 8(1):60–76, 1974.
23. Rohit Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.
24. Jean-Eric Pin and Pascal Weil. Ponynominal closure and unambiguous product. *Theory Comput. Syst.*, 30(4):383–422, 1997.
25. Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. In *Foundations of*

Software Technology and Theoretical Computer Science (FSTTCS), pages 363–375, 2013.
26. Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In *Mathematical Foundations of Computer Science (MFCS)*, pages 729–740, 2013.
27. Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 342–353, 2014.
28. Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *CoRR*, abs/1402.3277, 2014.
29. Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.
30. Imre Simon. Piecewise testable events. In H. Barkhage, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975.
31. Howard Straubing. Semigroups and languages of dot-depth two. *Theor. Comput. Sci.*, 58:361–378, 1988.
32. Lorijn van Rooijen and Marc Zeitoun. The separation problem for regular languages by piecewise testable languages. *CoRR*, abs/1303.2143, 2013.
33. Yechezkel Zalcstein. Locally testable languages. *J. Comput. Syst. Sci.*, 6(2):151–167, 1972.

Appendix

Some Scenarios for Separability in Practice

We have several practical motivations for studying separation problem.

One motivation for studying separability comes from a general interest in finding out when one can explain the differences between two complex structures (I and E) in a simple language (say, S). Such questions can be useful in various settings, for example:

(1) Query debugging, or explaining the result of queries in graph databases;
(2) Query approximation in the context of graph databases; and
(3) Debugging systems in a wider context.

We explain these scenarios next.
(1) In querying *graph databases* [4], one is given a database in the form of an edge-labeled graph. Typical queries that one evaluates over graph databases are, for example, *regular path queries (or RPQs)* [8] and its many variations; see e.g., [4] for an overview. An RPQ is simply a regular expression r that can be evaluated over the graph database G and returns all pairs of nodes (u, v) such that there exists a (not necessarily simple) path from u to v that is labeled by a word from $L(r)$. In this setting, separability can help, for example, to debug queries. When a given RPQ does not return a tuple (s, t) that one is expecting, a simple separator can describe the reason why. For example, if a separator includes the language of the RPQ and excludes the language of paths that lead from s to t, it describes why there exists no path from s to t that matches the RPQ, which, by definition of the usual semantics of RPQs, is precisely the reason why (s, t) is not in the answer. We note that evaluating RPQ-like queries on edge-labeled graphs is practically very relevant, e.g., for evaluation SPARQL 1.1 queries on RDF data [2, 18] and for Facebook's Graph Search [10], whose concepts are based on RPQ-like queries.

We note that, in this scenario, the RPQs and graphs are naturally abstracted as *regular word languages*, but more expressive query formalisms such as context-free languages have recently attracted interest as well [13].
(2) Graph databases may have index structures that allows them to evaluate certain operators more quickly than others. We can exploit separation to automatically generate a query that can be evaluated quickly and that sufficiently approximates the query we want to evaluate. For example, an expert could write two RPQs r^+ and r^- with the meaning that she wants the system to generate a separator s (that comes from a class of queries that makes efficient use of the index) and evaluate s on the graph data. Here, we should have that $L(r^+) \subseteq L(s)$ and $L(s) \cap L(r^-) = \emptyset$. Hence, the expert wants to obtain all results that match r^+. However, the data is huge and r^+ may be very expensive to evaluate. Therefore, she provides a second RPQ r^- which allows her to steer how closely she wants to approximate r^+. If the answer should be precise, she could choose r^- to define the complement language of r^+.

(3) Separation could also be used for debugging systems in a more general setting. For clarity, we describe an abstract scenario based on a simple finite-state system, but the principle is also valid in any other context.

Consider a finite-state system K. The system has an initial state and, from then on, moves to different states depending on its environment. At some point, however, the system does something strange and enters a state q^- which was not expected. Instead, we were expected the system to perform an action that is connected to some other state q^+. In order to understand what went wrong, it could make sense to try to understand the differences between $K^- = (Q, i, \{q^-\})$ and $K^+ = (Q, i, \{q^+\})$. Here, K^- is simply the system K in which we take q^- as accepting state; and analogously for K^+. Both systems describe a regular language: the language of actions that take K from its initial state i to state q^-, resp., q^+. If K is deterministic, these languages are disjoint. A separator could help to understand the reason why the system behaved in the way it did. For example, if $A^*aA^*bA^*bA^*$ would be a separator, it could for example mean that the system went to state q^- because it first received insufficient money (event a); and then its sensor for tilting gave two warnings (event b). Therefore, it went into some kind of panic mode instead of deploying coffee.

Undecidability of Deciding if a CFL is PT

Thomas Place pointed out the following observation to us. The proof we present is due to him and follows similar lines as the proof of Greibach's Theorem [12].

Observation 4. It is undecidable whether a given context-free language is piecewise testable.

Proof. Let A be the alphabet and L be a context-free language over A. We show that, if we can decide whether L is piecewise testable, we can also decide whether L is universal.

Fix K as any context-free language that is not piecewise-testable and $\#$ to be a symbol that is not in A. The (context-free) language $L' = K\#A^* \cup A^*\#L$ is piecewise-testable iff L is universal.

If L is universal, $L' = A^*\#A^*$ which is trivially piecewise.

If L is not universal, assume by contradiction that L' is piecewise. By hypothesis there is $w \notin L$. It is then immediate that $K = L'(\#w)^{-1}$ is also piecewise (piecewise languages are closed under residuals) which is a contradiction by choice of K.

Proofs for Observations

Observation 5. Every regular language is a labeled VAS language.

Proof. Let a regular language L be accepted by a deterministic finite automaton $\mathcal{A}$ with states Q, initial state $q_1 \in Q$, set of final states $F \subseteq Q$, alphabet A and

the set of transitions $\delta \subseteq Q \times A \times Q$. Let $Q = \{q_1, \ldots, q_k\}$. Intuitively $\mathcal{A}$ being in state q_i will be simulated by VAS M being in the marking

$$(0, \ldots, 0, 1, 0, \ldots, 0),$$

where the only 1 stays at the i-th coordinate. Formally, for every transition $(q_i, a, q_j) \in \delta$ we add the following vector u into the set of transitions T of VAS M:

$$(0, \ldots, 0, 1, 0, \ldots, 0, -1, 0, \ldots, 0),$$

where 1 stays at the i-th coordinate and -1 at the j-th coordinate (here $i < j$ only for illustration, it does not hold in general). Moreover then we label u by the letter a, $\ell(u) = a$. Additionally, for every $q_i \in F$ we add to the set T the following transition:

$$(0, \ldots, 0, -1, 0, \ldots, 0),$$

where -1 stays at the i-th coordinate. It is easy to see that the language of M with a source equal $(1, 0, \ldots, 0)$ and target equal $(0, \ldots, 0)$ equals L, thus indeed every regular language is also a language of a VAS.

Observation 6. VAS languages are closed under intersection.

Proof. We prove it using a standard product construction. Let $M_1 = (A, T_1, \ell_1, s_1, t_1)$ and $M_2 = (A, T_2, \ell_2, s_2, t_2)$ be k_1 and k_2 dimensional labeled VASs, respectively. For $v_1 \in \mathbb{Z}^{n_1}$ and $v_2 \in \mathbb{Z}^{n_2}$ vector $v_1 \circ v_2 \in \mathbb{Z}^{n_1+n_2}$ is a vector which has first n_1 coordinates like v_1 and next n_2 coordinates like v_2. Let $M = (A, T, \ell, s, t)$ be a $k_1 + k_2$ dimensional labeled VAS, such that,

$$T = \{t_1 \circ t_2 \mid t_1 \in T_1, t_2 \in T_2, \ell_1(t_1) = \ell_2(t_2)\},$$

source $s = s_1 \circ s_2$ and target $t = t_1 \circ t_2$. It is easy to see that $L(M) = L(M_1) \cap L(M_2)$.