Regular Separability in Büchi VASS

Pascal Baumann
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Roland Meyer
TU Braunschweig, Germany

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

We study the (ω-)regular separability problem for Büchi VASS languages: Given two Büchi VASS with languages L_1 and L_2, check whether there is a regular language that fully contains L_1 while remaining disjoint from L_2. We show that the problem is decidable in general and PSPACE-complete in the 1-dimensional case, assuming succinct counter updates. The results rely on several arguments. We characterize the set of all regular languages disjoint from L_2. Based on this, we derive a (sound and complete) notion of inseparability witnesses, non-regular subsets of L_1. Finally, we show how to symbolically represent inseparability witnesses and how to check their existence.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Separability problem, Vector addition systems, Infinite words, Decidability

Digital Object Identifier 10.4230/LIPIcs.STACS.2023.9

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2301.11242

Funding Funded by the European Union (ERC, FINABIS, 101077902). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. The second author was supported by the DFG project EDS@SYN: Effective Denotational Semantics for Synthesis.

1 Introduction

The separability problem asks, given languages L_1 and L_2, whether there exists a language R that *separates* L_1 and L_2, meaning $L_1 \subseteq R$ and $R \cap L_2 = \emptyset$. Here, R is constrained to be from a particular class $\mathcal{S}$ of admitted separators. Since safety verification of systems with concurrent components is usually phrased as an intersection problem for finite-word languages, and separators certify disjointness, deciding separability can be viewed as synthesizing safety certificates. Analogously, deciding separability for infinite-word languages is a way of certifying liveness. If $\mathcal{S}$ is the class of (ω-)regular languages, we speak of *regular separability*.

Separability problems have been studied intensively over the last few years. If the input languages are themselves regular and $\mathcal{S}$ is a subclass [36, 35, 34, 33, 37, 38, 30, 14], then separability generalizes the classical subclass membership problem. Moreover, separability for languages of infinite-state systems has received a significant amount of attention [16, 15, 13, 12, 9, 8, 11, 1, 44, 42, 10, 7]. Let us point out two prominent cases.

First, one of the main open problems in this line of research is whether regular separability is decidable for (reachability) languages of *vector addition systems with states* (VASS): A VASS consist of finitely many control states and a set of counters that can be *incremented* and *decremented*, but not tested for zero. Moreover, each transition is labeled by a word over the input alphabet. Here, a run is accepting if it reaches a final state with all counters being zero. While there have been several decidability results for subclasses of the VASS languages [16, 13, 12, 9, 8], the general case remains open. Second, a surprising result is that

if K and L are coverability languages of *well-structured transition systems* (WSTS), then K and L are separable by a regular language if and only if they are disjoint [13]. As VASS are one example of WSTS, this result also applies to their coverability languages.

Regular separability in Büchi VASS. In this paper, we study the regular separability problem for Büchi VASS. These are VASS that accept languages of *infinite words*. A run is accepting if it visits some final state infinitely often. Since no condition is placed on the counter values, Büchi VASS languages are an infinite-word analogue of finite-word *coverability languages*, where acceptance is defined by the reached state (not the counters). The *regular separability problem* is to decide, given Büchi VASS $\mathcal{V}_1$ and $\mathcal{V}_2$, whether there exists an ω-regular language R such that $L(\mathcal{V}_1) \subseteq R$ and $L(\mathcal{V}_2) \cap R = \emptyset$.

Our main results are that (i) regular separability for Büchi VASS is decidable, and that (ii) for one-dimensional Büchi VASS (i.e. those with a single counter) the problem is PSPACE-complete. Here, we assume that the counter updates are encoded in binary.

Given that Büchi VASS accept using final states and their transition systems are WSTS, one may suspect that there is an analogue of the aforementioned result for WSTS: Namely, that two languages of Büchi VASS are separable by an ω-regular language if and only if they are disjoint. We show that this is not the case: There are Büchi VASS $\mathcal{V}_1$ and $\mathcal{V}_2$ such that $L(\mathcal{V}_1)$ and $L(\mathcal{V}_2)$ are disjoint, but not separable by an ω-regular language. In fact, we show an even larger disparity between these two problems for WSTS in the infinite-word case: We exhibit a natural class of WSTS for which intersection is decidable but regular separability is not. Thus, regular separability for Büchi VASS requires significantly new ideas and involves several phenomena that do not occur for finite-word languages of VASS.

New phenomena and key ingredients. We first observe that we can assume one input language to be fixed, namely an infinite-word version D_n of the Dyck language. Then, following the *basic separator* approach from [16], we identify a small class $\mathcal{B}$ of ω-regular languages such that L is separable from D_n if and only if L is included in a finite union of sets from $\mathcal{B}$. Here, a crucial insight is that a Büchi automaton can guarantee disjointness from D_n without knowing exactly when the letter balance crosses zero. Note that a negative letter balance is the exact condition for non-membership in D_n. In contrast, in the finite word case, there are always separating automata that can tell when zero is crossed [16]. This insight is also key to the example differentiating disjointness and separability in Büchi VASS, and to the undecidability proof for certain WSTS despite decidable disjointness.

We then develop a decomposition of Büchi VASS languages into *finitely many* pieces, which are induced by what we call *profiles*. Inspired by Büchi automata, the idea of a profile is to fix the set of transitions that can and have to be taken infinitely often in a run. Finding the right generalization to Büchi VASS, however, turned out to be non-trivial. Our formulation refers to edges in the Karp-Miller graph, augmented by constraints that guarantee the existence of an accepting run. The resulting decomposition has properties similar to the decomposition of VASS languages into run ideals [28], which has been useful for previous separability procedures [16, 11].

We associate to each profile a system of linear inequalities and show that separability holds if and only if each of these systems is feasible. While this yields decidability, checking feasibility is not sufficient to obtain a PSPACE-upper bound in the one-dimensional case. Instead, we use Farkas' Lemma to obtain a dual system of inequalities so that separability fails if and only if one dual system is satisfiable. A solution to a dual system yields a pattern in the Karp-Miller graph, called *inseparability flower*, which witnesses inseparability.

Compared to prior witnesses for deciding properties of VASS languages (e.g. regularity [17], language boundedness [6], and other properties [2]), inseparability flowers are quite unusual: they contain a non-linear condition, requiring one vector to be a scalar multiple of another.

For one-dimensional Büchi VASS, the condition degenerates into a linear one. This allows us to translate inseparability flowers into particular runs in a two-dimensional VASS subject to additional linear constraints. Using methods from [4], this yields a PSPACE procedure.

Related work. It was already shown in 1976 that regular separability is undecidable for context-free languages [41, 25]. Over the last decade, there has been intense interest in deciding regular separability for subclasses of finite-word VASS reachability languages: The problem is decidable for (i) reachability languages of one-dimensional VASS [12], (ii) coverability languages of VASS [13], (iii) reachability languages of Parikh automata [8], and (iv) commutative reachability languages of VASS [9]. Moreover, decidability still holds if one input language is an arbitrary VASS language and the other is as in (i)-(iii) [16]. As discussed above, for finite-word coverability languages of WSTS, regular separability is equivalent to disjointness [13]. Moreover, the aforementioned undecidability for context-free languages has been strengthened to visibly pushdown languages [27]. To our knowledge, for languages of infinite words, separability has only been studied for regular input languages [32, 24].

Our result makes use of Farkas' Lemma to demonstrate the absence of what can be understood as a linear ranking function (on letter balances). There are precursors to this. In liveness verification [39], Farkas' Lemma has been used to synthesize, in a complete way, linear ranking functions proving the termination of while programs over integer variables. In the context of separability for finite words, Farkas' Lemma was used to distinguish separable from non-separable instances [16], similar to our approach. The novelty here is the combination of Farkas' Lemma with the new notion of profiles needed to deal with infinite runs.

The languages of Büchi VASS have first been studied by Valk [43] and (in the deterministic case) Carstensen [5]. Some complexity results (such as EXPSPACE-complexity of the emptiness problem) were shown by Habermehl [23]. More recently, there have been several papers on the topological complexity of Büchi VASS languages (and restrictions) [21, 18, 22]. See the recent article by Finkel and Skrzypczak [22] for an overview.

2 Preliminaries

Dyck Language. We use an infinite-word version of the Dyck language over n pairs of matching letters $a_i, \bar{a}_i$. We denote the underlying alphabet by $\Sigma_n := \bigcup_{i=1}^{n} \{a_i, \bar{a}_i\}$. The *Dyck language* contains those infinite words where every occurrence of $\bar{a}_i$ has a matching occurrence of a_i to its left: $D_n := \{w \in \Sigma_n^\omega \mid \forall v \in \mathrm{prefix}(w) \colon \forall i \in [1,n] \colon \varphi_i(v) \geq 0\}$. Here, $\varphi_i : \Sigma_n^* \to \mathbb{Z}$ is the ith (*letter*) *balance* function that computes for a given word w the difference $|w|_{a_i} - |w|_{\bar{a}_i}$. We also use $\varphi(w)$ for the vector $(\varphi_1(w), \ldots, \varphi_n(w)) \in \mathbb{Z}^n$.

Büchi VASS and Automata. A *Büchi vector addition system with states (Büchi VASS)* of dimension $d \in \mathbb{N}$ over alphabet Σ is a tuple $\mathcal{V} = (Q, q_0, T, F)$ consisting of a finite set of states Q, an initial state $q_0 \in Q$, a set of final states $F \subseteq Q$, and a finite set of transitions $T \subseteq Q \times \Sigma^* \times \mathbb{Z}^d \times Q$. The size of the Büchi VASS is $|\mathcal{V}| := |Q| + 1 + |F| + \sum_{(q,w,\delta,q') \in T} |w| + \sum_{i=1}^{d} \max\{\log |\delta(i)|, 1\}$. If $d = 0$, we call $\mathcal{V}$ a *Büchi automaton*.

The semantics of the Büchi VASS is defined over *configurations*, which are elements of $Q \times \mathbb{N}^d$. We call the second component in a configuration the *counter valuation* and refer to the entry in dimension i as the *value of counter* i. The *initial configuration* is $(q_0, \mathbf{0})$. We lift

the transitions of the Büchi VASS to a relation over configurations $\to \subseteq Q \times \mathbb{N}^d \times \Sigma^* \times Q \times \mathbb{N}^d$ as follows: $(q, \boldsymbol{m}) \xrightarrow{w} (q', \boldsymbol{m}')$ if there is $(q, w, \delta, q') \in T$ so that $\boldsymbol{m}' = \boldsymbol{m} + \delta$. A *run* of the Büchi VASS is an infinite sequence of transitions of the form $(q_0, \mathbf{0}) \xrightarrow{w_1} (q_1, \boldsymbol{m}_1) \xrightarrow{w_2} \cdots$ Thus, the sequence starts in the initial configuration and makes sure the target of one transition is the source of the next. The run is *accepting* if it visits final states infinitely often, meaning there are infinitely many configurations $(q, \boldsymbol{m})$ with $q \in F$. The run is said to be *labeled* by the word $w = w_0 w_1 \cdots$ in Σ^ω. The *language* $L(\mathcal{V})$ of the Büchi VASS consists of all infinite words that label an accepting run. Note that we can always ensure that every accepting run has an infinite-word label, by tracking in the state whether a non-ε-transition has occurred since the last visit to a final state. An infinite-word language is (ω-)*regular*, if it is the language of a Büchi automaton. As we only consider infinite-word languages, we just call them languages.

Karp-Miller Graphs. We work with the Karp-Miller graph $\mathsf{KM}(\mathcal{V})$ associated with a Büchi VASS $\mathcal{V}$ [26]. Since we are interested in infinite runs, we define the Karp-Miller graph as a Büchi automaton. Its state set is a finite set of *extended configurations*, which are elements of $Q \times (\mathbb{N} \cup \{\omega\})^d$. The initial state is the initial configuration in the Büchi VASS. The final states are those extended configurations $(q, \boldsymbol{m})$ with $q \in F$. The transitions are labeled by T, so instead of letters they carry full Büchi VASS transitions. An entry ω in an extended configuration denotes the fact that a prefix of a run can be repeated to produce arbitrarily high counter values. More precisely, the Karp-Miller graph is constructed as follows. From an extended configuration $(q, \boldsymbol{m})$ we have a transition labeled by (q_1, a, δ, q_2), if $q = q_1$ and $\boldsymbol{m} + \delta$ remains non-negative. The latter addition is defined componentwise and assumes $\omega + k := \omega =: k + \omega$ for all $k \in \mathbb{Z}$. The result of taking the transition is the extended configuration $(q_2, \boldsymbol{m}_2)$, where $\boldsymbol{m}_2$ is constructed from $\boldsymbol{m} + \delta$ as follows. We raise to ω all counters i for which there is an earlier configuration $(q_2, \boldsymbol{m}_1)$ with $\boldsymbol{m}_1 \leq \boldsymbol{m} + \delta$ and $\boldsymbol{m}_1(i) < [\boldsymbol{m} + \delta](i)$, earlier meaning on some path from $(q_0, \mathbf{0})$ to $(q, \boldsymbol{m})$. If this is the case, the path from $(q_2, \boldsymbol{m}_1)$ to $(q_2, \boldsymbol{m} + \delta)$ can be repeated indefinitely to produce arbitrarily high values for counter i. We refer to the repetition of such a path in a run as *pumping*.

The Karp-Miller graph over-approximates the language of the Büchi VASS in the following sense. Every infinite sequence of transitions that leads to a run of the Büchi VASS is the labeling of an infinite run in the Karp-Miller graph. Moreover, if the run of the Büchi VASS is accepting, so is the run in the Karp-Miller graph. In the other direction, every finite transition sequence in the Karp-Miller graph represents a transition sequence in the Büchi VASS. The sequence in the Büchi VASS, however, may be longer to compensate negative effects on ω-entries by pumping.

3 Problem, Main Result, and Proof Outline

A language R is a *regular separator* for a pair of languages L_1, L_2, if R is regular, $L_1 \subseteq R$, and $R \cap L_2 = \emptyset$. We write $L_1 \,|\, L_2$ for the fact that a regular separator exists. We consider here languages of Büchi VASS, and formulate the *regular separability problem* as follows. Given Büchi VASS $\mathcal{V}_1$, $\mathcal{V}_2$, check whether $L(\mathcal{V}_1) \,|\, L(\mathcal{V}_2)$ holds. Our main result is the following.

▶ **Theorem 3.1.** *The regular separability problem for Büchi VASS is decidable.*

It should be noted that our procedure is non-primitive recursive, as it explicitly constructs the Karp-Miller graph of an input Büchi VASS, which can be of Ackermannian size [31, Theorem 2]. In the case of VASS coverability languages (and even for more general WSTS),

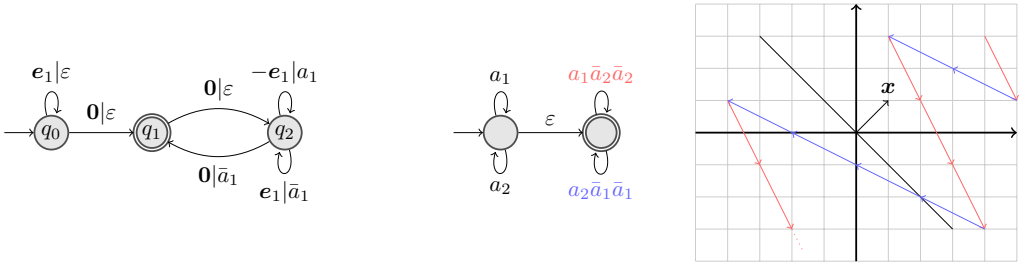


Figure 1 Left: A Büchi VASS accepting a language S with $S \cap D_1 = \emptyset$ but $S \not| D_1$. Here, $e_1 \in \mathbb{Z}$ is the one-dim. vector with entry 1. Right: A regular language that is not included in a finite union of languages $P_{i,k}$ and $S_{i,k}$, but that is included in $S_{\boldsymbol{x},k}$ for $\boldsymbol{x} = (1,1)$, $k = 1$. The horizontal and vertical dimensions denote the balance for a_1 resp. a_2.

it is known that regular separability is equivalent to disjointness [13]. Thus, for finite words, separability reduces to the much better understood problem of disjointness. For the infinite-word languages considered here, the situation is different.

▶ **Theorem 3.2.** *There are Büchi VASS languages L_1, L_2 with $L_1 \cap L_2 = \emptyset$ and $L_1 \not| L_2$. There are classes of WSTS where intersection is decidable but separability is not.*

For the second statement, we introduce the class of *weak Büchi reset VASS*, which are VASS with reset instructions, with the additional constraint that each run can only use resets a finite number of times.

For the first statement of Theorem 3.2, let us give an intuition. We choose $L_1 = L(\mathcal{V})$, where $\mathcal{V}$ is the Büchi VASS in Figure 1(left), and $L_2 = D_1$, the Dyck language. To show $L(\mathcal{V}) \not| D_1$, suppose there is a Büchi automaton $\mathcal{A}$ with n states such that $L(\mathcal{V}) \subseteq L(\mathcal{A})$ and $L(\mathcal{A}) \cap D_1 = \emptyset$. Then $\mathcal{A}$ has to accept $(a_1^n \bar{a}_1^{n+1})^\omega \in L(\mathcal{V})$. However, pumping yields that for some $m > n$ the word $(a_1^m \bar{a}_1^{n+1})^\omega \in D_1$ also has to be accepted by $\mathcal{A}$, contradiction. Moreover, to show $L(\mathcal{V}) \cap D_1 = \emptyset$ we observe that in accepting runs of $\mathcal{V}$, almost every visit (meaning: all but finitely many) to the final state drops the letter balance by 1. Therefore on any accepting run this balance eventually becomes negative, yielding a word outside of D_1.

In the remainder of the section, we outline the proof of Theorem 3.1. Assume we are given $L_1 = L(\mathcal{V}_1)$ and $L_2 = L(\mathcal{V}_2)$ and this is a non-trivial instance of separability, meaning L_1, L_2 are not regular and $L_1 \cap L_2 = \emptyset$. For proving separability, we could enumerate regular languages until we find a separator. The difficult part is disproving separability. Inseparability of L_1 and L_2 is witnessed by a set of words $W \subseteq L_1$ so that every regular language R containing them already intersects L_2, formally: $W \subseteq R$ implies $R \cap L_2 \neq \emptyset$. Showing the existence of such a set W is difficult for two reasons. First, it is unclear which sets of words ensure the universal quantification over all regular languages. Second, as we have a non-trivial instance of separability, W (if it exists) will be a non-regular language. So it is unclear how to represent it in a finite way and how to check its existence.

To address the first problem and understand the sets of words that disprove separability, we use diagonalization. Call an *(L_2-)separator candidate* a regular language that is disjoint from L_2. Let $R_1, R_2, \ldots$ be an enumeration of the separator candidates. If L_1 is not separable from L_2, for every R_i there is a word $w_i \in L_1$ with $w_i \notin R_i$. We call such a set of words $W = \{w_1, w_2, \ldots\}$ that escapes every separator candidate an *inseparability witness*.

▶ **Observation 3.3.** *$L_1 \not| L_2$ if and only if there is an inseparability witness.*

Our decision procedure will check the existence of an inseparability witness. We obtain the procedure in four steps: the first is a simplification, the second is devoted to understanding the separator candidates, the third is another simplification, and the last characterizes the inseparability witnesses and checks their existence.

Step 1: Fixing L_2. We first reduce general regular separability to regular separability from the Dyck language. The reduction is simple and works just as for finite words [16].

▶ **Lemma 3.4.** *Given Büchi VASS $\mathcal{V}_1$ and $\mathcal{V}_2$, we can compute a Büchi VASS $\mathcal{V}$ over Σ_n so that $L(\mathcal{V}_1) \,|\, L(\mathcal{V}_2)$ if and only if $L(\mathcal{V}) \,|\, D_n$, where n is the dimension of $\mathcal{V}_2$.*

Step 2: Understanding the Separator Candidates. To understand the regular languages that are disjoint from D_n, we will define *basic separators*, sets $P_{i,k}$ and $S_{\boldsymbol{x},k}$, on which we elaborate in a moment. The following theorem says that finite unions of basic separators are sufficient for regular separability. This is our first technical result and shown in Section 4.

▶ **Theorem 3.5.** *If $R \subseteq \Sigma_n^\omega$ is regular and $R \cap D_n = \emptyset$, then R is included in a finite union of basic separators.*

For the definition of $P_{i,k}$, we note that the words outside D_n have, for some index $i \in [1, n]$, an earliest moment in time where the balance between a_i and $\bar{a}_i$ falls below zero. To turn this into a regular language, we impose an upper bound $k \in \mathbb{N}$ on the (positive) balance between the letters a_i and $\bar{a}_i$ that is maintained until the earliest moment is reached. This yields the regular language

$$P_{i,k} \;:=\; \{w \in \Sigma_n^\omega \mid \exists v \in \mathrm{prefix}(w) \colon \varphi_i(v) < 0 \wedge \forall u \in \mathrm{prefix}(v) \colon \varphi_i(u) \leq k\}.$$

The family of languages $P_{i,k}$ already captures the complement of D_n. The problem is that we may need infinitely many such languages to cover the language R of interest. For every bound k, a regular R with $R \cap D_1 = \emptyset$ may contain a word with a higher balance before falling below zero, take for example $R = a_1^* \bar{a}_1^\omega$. The first insight is that if R can fall below zero from arbitrarily high values, then the underlying Büchi automaton has to contain loops with a negative balance. The R thus contains words uv with an unconstrained prefix and a suffix that decomposes into $v = v_1 v_2 \cdots$ so that every infix $w = v_\ell$ has a negative balance on letter a_i. The observation suggests the definition of a language that contains precisely the words $u.v$. To make the language regular, we impose a bound k on the positive balance that can be used during the infixes w. Call the resulting language $S_{i,k}$. Unfortunately, taking the $P_{i,k}$ and the $S_{i,k}$ as basic separators is still not enough: Figure 1(right) exhibits a regular language, disjoint from D_1, that is not included in a finite union of $P_{i,k}$ and $S_{i,k}$, because it contains infixes where the balance on each letter exceeds all bounds in each coordinate.

The second insight is that we can catch the remaining words with a version of $S_{i,k}$ that weights coordinates with some $\boldsymbol{x} \in \mathbb{N}^n$. Let us give some intuition on this. The words from R that we cannot catch with a $P_{i,k}$ must come across, for each i that becomes negative, a loop with positive balance on i (otherwise, the balance on those i would be bounded). But then, the only way such words can avoid D_1 is by ending up in a strongly connected component where *every* loop (with a final state) makes progress towards crossing 0, i.e. is negative in some coordinate. One can then conclude that even all $\mathbb{Q}_{\geq 0}$-linear combinations of loops (a convex set) must avoid the positive orthant $\mathbb{Q}^n_{\geq 0} \subset \mathbb{Q}^n$. By the Hyperplane Separation Theorem (we use it in the form of Farkas' Lemma), this is certified by a hyperplane that separates all loop effects from $\mathbb{Q}^n_{\geq 0}$. This hyperplane is given by some orthogonal vector $\boldsymbol{x} \in \mathbb{N}^n$, meaning that every loop balance must have negative scalar product with $\boldsymbol{x}$. Hence, we can catch these words by:

$$S_{\boldsymbol{x},k} := \left\{ u.v \in \Sigma_n^\omega \;\middle|\; \begin{array}{l} \text{a.) } \forall f \in \mathrm{infix}(v)\colon \langle \boldsymbol{x}, \varphi(f)\rangle \leq k, \text{ and} \\ \text{b.) } v = v_0.v_1.v_2 \cdots \wedge \forall \ell \in \mathbb{N}\colon \langle \boldsymbol{x}, \varphi(v_\ell)\rangle < 0 \end{array} \right\}.$$

Coming back to Figure 1(right), the weight vector $\boldsymbol{x} = (1,1)$ guarantees that the weighted balance decreases indefinitely and also the weighted balances of all infixes stay bounded. In [16], a similar argument has been used to show sufficiency of basic separators.

Step 3: Pumpable Languages. With the basic separators at hand, the task is to understand the sets of words witnessing inseparability. While studying this problem, we observed that the argumentation for the $P_{i,k}$ was always similar to the one for the $S_{\boldsymbol{x},k}$. This led us to the question of whether we can get rid of the $P_{i,k}$ in separators. The answer is positive, and hinges on a new notion of pumpability for languages over Σ_n.

Call infinite words u and v *equivalent*, written $u \sim v$, if v can be obtained from u by removing and inserting finitely many letters: There are $u_0, v_0 \in \Sigma^*$ and $w \in \Sigma^\omega$ such that $u = u_0 w$ and $v = v_0 w$. We say that a language $L \subseteq \Sigma_n^\omega$ is *pumpable* if for every $w \in L$ and every $k \in \mathbb{N}$, there exists a decomposition $w = w_0 w_1$ and a word $w_0' \in \Sigma_n^*$ that is a prefix of a word in D_n such that $w_0'.w_1 \in L$ and the letter balance satisfies the following: (a) $\varphi(w_0') \geq \varphi(w_0)$ and (b) for the indices $i \in [1,n]$ where φ_i becomes negative on some prefix of w, we have $\varphi_i(w_0') \geq \max\{\varphi_i(w_0), 0\} + k$. The consequence of this definition is that a pumpable language leaves every language $P_k := \bigcup_{i\in[1,n]} P_{i,k}$. Indeed, for every word $w \in L$ and every $k \in \mathbb{N}$, there is a word $w' \in L$ with $w \sim w'$ where the letter balance exceeds k before becoming negative, and thus $w' \notin P_k$. With the previous characterization of separator candidates, what is left to separate L from D_n are the languages $S_{\boldsymbol{x},k}$.

▶ **Lemma 3.6.** *If $L \subseteq \Sigma_n^\omega$ is pumpable, then $L \mid D_n$ if and only if $L \mid\!\!\!\!_{\lim} D_n$, where $L \mid\!\!\!\!_{\lim} D_n$ means $L \subseteq \bigcup_{\boldsymbol{x}\in X} S_{\boldsymbol{x},k}$ for some finite set $X \subseteq \mathbb{N}^n$ and some $k \in \mathbb{N}$.*

In our context, pumpability is interesting because we can turn every Büchi VASS language into a pumpable language without affecting separability.

▶ **Theorem 3.7.** *Let $\mathcal{V}$ be a d-dim. Büchi VASS over Σ_n. We can compute a d-dim. Büchi VASS $\mathcal{V}_{\mathsf{pump}}$ that satisfies the following:*

1. *$L(\mathcal{V}_{\mathsf{pump}})$ is pumpable,*
2. *there is a $k \in \mathbb{N}$ so that $L(\mathcal{V}_{\mathsf{pump}}) \subseteq L(\mathcal{V}) \subseteq L(\mathcal{V}_{\mathsf{pump}}) \cup P_k$, and*
3. *$L(\mathcal{V}) \mid D_n$ if and only if $L(\mathcal{V}_{\mathsf{pump}}) \mid D_n$.*

The construction of $\mathcal{V}_{\mathsf{pump}}$ employs the Karp-Miller graph in an original way, namely to track the unboundedness of letter balances. Let $\bar{\mathcal{V}}$ be the $(d+n)$-dimensional Büchi VASS obtained from $\mathcal{V}$ by tracking the effect of the letters from Σ_n in n additional counters. For $\bar{\mathcal{V}}$, we construct the Karp-Miller graph. The relationship between the languages of $\mathsf{KM}(\bar{\mathcal{V}})$ and $\mathcal{V}$ is as follows. For all words where every letter balance stays non-negative, their runs in $\mathcal{V}$ can be mimicked in $\mathsf{KM}(\bar{\mathcal{V}})$. For all other words, where the balance eventually becomes negative, this only holds if the corresponding counter in $\bar{\mathcal{V}}$ has been raised to ω beforehand. Essentially, the new Büchi VASS $\mathcal{V}_{\mathsf{pump}}$ restricts $\mathcal{V}$ to those runs that have counterparts in $\mathsf{KM}(\bar{\mathcal{V}})$. This is achieved with a simple product construction of $\mathcal{V}$ and $\mathsf{KM}(\bar{\mathcal{V}})$. The thing to note is that every word from $L(\mathcal{V})$ that does not make it into $L(\mathcal{V}_{\mathsf{pump}})$ belongs to P_k, where k is the maximum concrete number in $\mathsf{KM}(\bar{\mathcal{V}})$: A run in $\mathcal{V}$ that cannot be mimicked in $\mathsf{KM}(\bar{\mathcal{V}})$ will at some point have a negative letter balance, before reaching ω in $\mathsf{KM}(\bar{\mathcal{V}})$ in that component; thus all counter values had been at most k until that point.

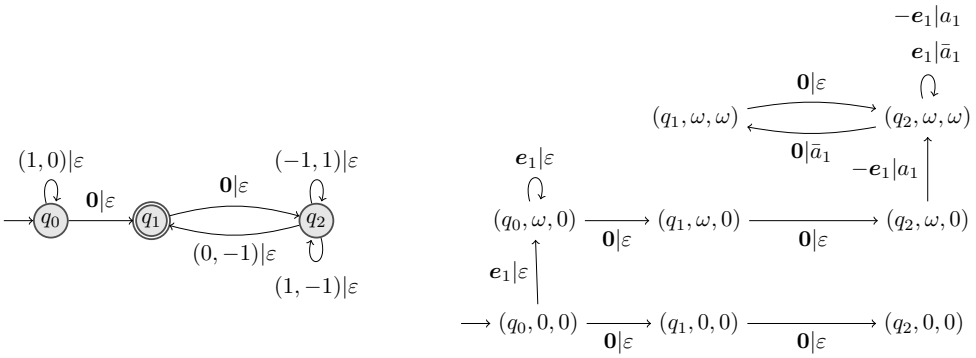


Figure 2 Left: The Büchi VASS $\bar{\mathcal{V}}$ constructed from the Büchi VASS $\mathcal{V}$ found in Figure 1(left). Note how the added second counter tracks the letter balance of the now removed transition labels, incrementing on letter a_1 and decrementing on letter $\bar{a}_1$. Right: The Büchi VASS $\mathcal{V}_{\mathsf{pump}}$ corresponding to $\mathcal{V}$ as given by Theorem 3.7. Here we did not mark the final states to reduce visual clutter; every state that includes q_1 is considered final. Similarly, the two labels above the loop in the top right correspond to two distinct transitions. Note that $\mathcal{V}_{\mathsf{pump}}$ essentially looks like $\mathsf{KM}(\bar{\mathcal{V}})$, just with different transition labels.

An example on how to construct $\bar{\mathcal{V}}$ and $\mathcal{V}_{\mathsf{pump}}$ can be found in Figure 2, where both were constructed for the Büchi VASS found in Figure 1(left).

In the proof of Theorem 3.5, we make use of Theorem 3.7 (recall that a regular language is the language of a 0-dimensional Büchi VASS). This may look like cyclic reasoning, but it is not: We will show Theorem 3.7(1)+(2) directly, using the arguments above. With this, we prove Theorem 3.5, which in turn is used to derive Lemma 3.6 and Theorem 3.7(3).

Step 4: Non-Separability Witnesses and Decidability. Because of pumpability, it remains to decide whether a Büchi VASS language $L(\mathcal{V})$ is included in a finite union $\bigcup_{\boldsymbol{x}\in X} S_{\boldsymbol{x},k}$ for some k. Part of the difficulty is that we have no bound on the cardinality of X. To circumvent this, we decompose $L(\mathcal{V})$ into a finite union $\bigcup_\pi L_\pi(\mathcal{V})$, where π is a *profile*, meaning a set of edges in $\mathsf{KM}(\mathcal{V})$ seen infinitely often during a run of $\mathcal{V}$. We then show that each $L_\pi(\mathcal{V})$ is either (i) included in a single separator $S_{\boldsymbol{x},k}$ or (ii) escapes every finite union $\bigcup_{\boldsymbol{x}\in X} S_{\boldsymbol{x},k}$.

Here, it is key to show an even stronger fact: In case (i), not only $L_\pi(\mathcal{V})$ is included in some $S_{\boldsymbol{x},k}$, but the entire set of runs in $\mathsf{KM}(\mathcal{V})$ that eventually remain in π. The advantage of strengthening is that finiteness of $\mathsf{KM}(\mathcal{V})$ allows us to express inclusion in $S_{\boldsymbol{x},k}$, for some k, as a *finite* system of linear inequalities over $\boldsymbol{x}$: We say that (1) the balance of every primitive cycle, weighted by $\boldsymbol{x}$, is at most zero and (2) the balance, weighted by $\boldsymbol{x}$, of some cycle containing all edges from π is negative. Here, (1) and (2) correspond to Conditions a.) and b.) of $S_{\boldsymbol{x},k}$. If they are met, then the runs of $\mathsf{KM}(\mathcal{V})$ along π are included in $S_{\boldsymbol{x},k}$ for some k.

We then prove that if the system is not feasible, then $\mathcal{V}$ has runs that escape every finite union $\bigcup_{\boldsymbol{x}\in X} S_{\boldsymbol{x},k}$. To this end, we employ Farkas' Lemma: It tells us that if there is no solution, then the dual system has a solution. The solution of the dual system can be interpreted as an executable linear combination of primitive cycles with non-negative balances. We show that these cycles can be arranged in a pattern in $\mathsf{KM}(\mathcal{V})$ we call *inseparability flower*. Such an inseparability flower then yields a sequence of runs $\rho_1, \rho_2, \ldots$ in $\mathsf{KM}(\mathcal{V})$ such that ρ_k escapes $S_{\boldsymbol{x},k}$ for every vector $\boldsymbol{x}$. Finally, pumpability allows us to lift these runs of $\mathsf{KM}(\mathcal{V})$ to runs of $\mathcal{V}$ and thus conclude inseparability.

This equips us with two possible decision procedures: We can either check solvability of each system of inequalities, or detect inseparability flowers in $\mathsf{KM}(\mathcal{V})$.

4 Basic Separators

We prove Theorem 3.5, that any regular language R over Σ_n with $R \cap D_n = \emptyset$ is contained in a finite union of languages $P_{i,k}$ and $S_{\boldsymbol{x},k}$. Note that a single value of k is sufficient, since we have $P_{i,k} \subseteq P_{i,k+1}$ and $S_{\boldsymbol{x},k} \subseteq S_{\boldsymbol{x},k+1}$ for each $i, \boldsymbol{x}, k$. The proof decomposes the Büchi automaton for R in a way that allows us to forget about connectedness issues and reason over cycles (and their letter balances) using techniques from linear algebra. We make use of the following basic fact from linear programming [40, Corollary 7.1f].

▶ **Theorem 4.1** (Farkas' Lemma (variant), [40]). *Let $\boldsymbol{A} \in \mathbb{Q}^{m \times n}$ be a matrix and let $\boldsymbol{b} \in \mathbb{Q}^m$ be a vector. Then the system $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ has a solution $\boldsymbol{x} \in \mathbb{Q}^n_{\geq 0}$ if and only if $\boldsymbol{y}^\top \boldsymbol{b} \geq 0$ for each vector $\boldsymbol{y} \in \mathbb{Q}^m_{\geq 0}$ with $\boldsymbol{y}^\top \boldsymbol{A} \geq \boldsymbol{0}$.*

Decomposing with profiles. We decompose $R = L(\mathcal{A})$ into a (not necessarily disjoint) union of several languages, each linked to a so-called *profile*. We will later see that for pumpable R, every such profile language already has to be contained in a single $S_{\boldsymbol{x},k}$.

▶ **Definition 4.2.** *Let $\mathcal{A}$ be a Büchi automaton. A* profile of $\mathcal{A}$ *is a set π of transitions of $\mathcal{A}$ for which there exists a cycle σ_π in $\mathcal{A}$ such that (a) σ_π contains exactly the transitions in π, and (b) σ_π starts (and ends) in a final state q_π.*

We denote by $\Pi(\mathcal{A})$ the finite set of profiles of $\mathcal{A}$. Moreover, we associate to every accepting run ρ of $\mathcal{A}$ its profile $\Pi(\rho)$, which contains exactly the transitions appearing infinitely often in ρ. This definition is sound, as the infinitely occurring transitions of an accepting run must form a cycle due to repetition, which visits a final state due to acceptance.

Given a profile π of $\mathcal{A}$, we define $L_\pi(\mathcal{A}) \subseteq L(\mathcal{A})$ to be the language of all words that have an accepting run ρ of $\mathcal{A}$ with $\Pi(\rho) = \pi$. Note that this language is still regular: From $\mathcal{A}$ one can construct a Büchi automaton that guesses a point after which only transitions from π can occur, and once this point is reached it keeps a list of already used transitions from π in each state. Then only once all transitions of π have been used the state becomes final and the list is set back to empty.

This now allows us to view R as the union of the languages $L_\pi(\mathcal{A})$ with $\pi \in \Pi(\mathcal{A})$. We show that each language $L_\pi(\mathcal{A})$ is either contained in $S_{\boldsymbol{x},k}$ for some $\boldsymbol{x}, k$, or there is a cycle that, assuming the pumpability from the previous section, makes $L_\pi(\mathcal{A})$ intersect D_n.

▶ **Lemma 4.3.** *Let $\mathcal{A}$ be a Büchi automaton over Σ_n and let π be one of its profiles. Then one of the following conditions holds:*

- **(i)** *There is a number $k \in \mathbb{N}$ and a vector $\boldsymbol{x} \in \mathbb{N}^n$ such that $L_\pi(\mathcal{A}) \subseteq S_{\boldsymbol{x},k}$, or*
- **(ii)** *there is a cycle σ' in $\mathcal{A}$ over w' with $\varphi(w') \geq \boldsymbol{0}$, and σ' contains all transitions from π.*

Assume $L_\pi(\mathcal{A}) \neq \emptyset$, otherwise Condition (i) trivially holds. We build a system $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ of linear inequalities as follows. It contains one inequality $\langle \boldsymbol{x}, \varphi(v) \rangle \leq 0$ for each word v read by a primitive cycle of transitions in π. By *primitive cycle* we mean a cycle that does not repeat a state. Moreover, the system contains the inequality $\langle \boldsymbol{x}, \varphi(v_\pi) \rangle \leq -1$ for the cycle σ_π over v_π that justifies the profile π. Let us quickly remark that the solution space of the system $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ is independent of the precise choice of the justifying cycle σ_π: To see this, we claim that $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ holds if and only if all primitive cyles in π have an $\boldsymbol{x}$-weighted balance at most zero, and at least one primitive cycle in π has a strictly negative $\boldsymbol{x}$-weighted balance. For the "if" direction, note that a sufficiently long repetition of σ_π will contain each primitive cycle as a (possibly non-contiguous) subsequence. This means, the repetition, and thus σ_π,

must have a strictly negative $\boldsymbol{x}$-weighted balance. For the converse, we observe that σ_π can be decomposed into primitive cycles. Thus, if σ_π has strictly negative $\boldsymbol{x}$-weighted balance, then so must at least one of its constituent primitive cycles.

Applying Farkas' Lemma to $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ either yields a solution $\boldsymbol{x} \in \mathbb{Q}^n_{\geq 0}$ or a vector $\boldsymbol{y} \in \mathbb{Q}^m_{\geq 0}$ with $\boldsymbol{y}^\top \boldsymbol{A}_\pi \geq \boldsymbol{0}$ and $\boldsymbol{y}^\top \boldsymbol{b} < 0$. In both cases we assume wlog. that the given vector has entries in $\mathbb{N}$, as we can always multiply with the lcm of the denominators.

Suppose we have a solution $\boldsymbol{x}$. We claim that then $L_\pi(\mathcal{A}) \subseteq S_{\boldsymbol{x},k}$, where $k = |Q_\pi| \cdot h$ and h is the maximal length of a transition label of $\mathcal{A}$. This is because $\boldsymbol{x}$ weights primitive cycles non-positively, and k is chosen such that for any infix v of a word in $L_\pi(\mathcal{A})$, if $|v| > k$, then v's associated transition sequence has to contain a primitive cycle. Thus, infixes at almost all start positions of a word in $L_\pi(\mathcal{A})$ must have $\boldsymbol{x}$-weighted balance $\leq k$.

If we obtain a vector $\boldsymbol{y} = (y_1, \ldots, y_m)$, then we can view it as a selection of rows in the matrix $\boldsymbol{A}_\pi$, where the jth row is being selected y_j many times. Since each row corresponds to a cycle, this is also a selection of cycles. Then by $\boldsymbol{y}^\top \boldsymbol{b} < 0$ we selected σ_π, where we can insert the other selected cycles. By $\boldsymbol{y}^\top \boldsymbol{A}_\pi \geq \boldsymbol{0}$ this forms a cycle σ' as required, with non-negative letter balance for all letter pairs.

Here, we used a system of linear inequalities $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$, which was solely dependent on $\mathcal{A}$ and π. We reasoned that if this system has a solution, then Condition (i) has to hold. This is a fact that we want to refer to in a later proof, and therefore we formalize it here.

▶ **Corollary 4.4.** *If $\mathcal{A}$ is a Büchi automaton with a profile π for which there is an $\boldsymbol{x} \in \mathbb{N}^n$ with $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$, then $L_\pi(\mathcal{A}) \subseteq S_{\boldsymbol{x},k}$ for some $k \in \mathbb{N}$.*

With Theorem 3.7 and Lemma 4.3, we can now show Theorem 3.5. Suppose $R = L(\mathcal{A})$ for some Büchi automaton $\mathcal{A}$. First, applying Theorem 3.7 with $d = 0$ yields a Büchi automaton $\mathcal{A}_{\mathsf{pump}}$ such that $L(\mathcal{A}) \subseteq L(\mathcal{A}_{\mathsf{pump}}) \cup P_\ell$ for some $\ell \in \mathbb{N}$ and $L(\mathcal{A}_{\mathsf{pump}}) \cap D_n = \emptyset$. Therefore, it suffices to show that $L(\mathcal{A}_{\mathsf{pump}})$ is included in a finite union of languages $S_{\boldsymbol{x},k}$. Suppose not. Then the set $L(\mathcal{A}_{\mathsf{pump}})$ decomposes into the sets $L_\pi(\mathcal{A}_{\mathsf{pump}})$ for $\pi \in \Pi(\mathcal{A}_{\mathsf{pump}})$. By Lemma 4.3, we know that for some π, Condition (ii) must hold: Otherwise, each $L_\pi(\mathcal{A}_{\mathsf{pump}})$ would be included in some $S_{\boldsymbol{x},k}$. But if (ii) holds for π, then there is a cycle σ' in $\mathcal{A}_{\mathsf{pump}}$ that contains π (and thus visits a final state) and reads a word v with $\varphi(v) \geq \boldsymbol{0}$. Now for some finite prefix u, the word uv^ω belongs to $L(\mathcal{A}_{\mathsf{pump}})$. Since $\varphi(v) \geq \boldsymbol{0}$, there is some lower bound $B \in \mathbb{Z}$ such that for each $i \in [1, n]$ and every prefix p of uv^ω, we have $\varphi_i(p) \geq B$. Finally, since $L(\mathcal{A}_{\mathsf{pump}})$ is pumpable, we can exchange a prefix in $w = uv^\omega$ to obtain another word $w' \in L(\mathcal{A}_{\mathsf{pump}})$ where every prefix p has $\varphi(p) \geq \boldsymbol{0}$. Hence $w' \in D_n$ and thus $L(\mathcal{A}_{\mathsf{pump}}) \cap D_n \neq \emptyset$, a contradiction.

5 Deciding Regular Separability

We now present the algorithm to decide, given a Büchi VASS $\mathcal{V}$ whether $L(\mathcal{V}) \mid D_n$. We first employ Theorem 3.7, because for pumpable languages we only have to deal with one type of basic separators. The next step is to generalize the notion of profiles from Büchi automata to Büchi VASS. Recall that for a sequence χ of transitions in $\mathcal{V}$, $\delta(\chi)$ denotes its effect on the counters of $\mathcal{V}$. If χ is a transition sequence in $\mathsf{KM}(\mathcal{V})$, then χ is labeled with a transition sequence of $\mathcal{V}$, so we define $\delta(\chi)$ accordingly. Since we consider Büchi VASS with input alphabet Σ_n, we write $\varphi(\chi)$ for the image of the input word under φ. Again, this notation is used for transition sequences in $\mathsf{KM}(\mathcal{V})$. We also write $\Delta(\chi) = (\delta(\chi), \varphi(\chi))$.

▶ **Definition 5.1.** *Let $\mathcal{V}$ be a Büchi VASS. A* profile for $\mathcal{V}$ *is a set π of edges in $\mathsf{KM}(\mathcal{V})$ for which there exists a cycle σ in $\mathsf{KM}(\mathcal{V})$ such that (i) σ contains exactly the edges in π, (ii) σ starts (and ends) in a final state, and (iii) $\delta(\sigma) \geq \boldsymbol{0}$.*

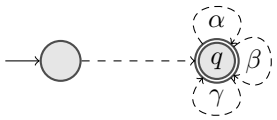
Clearly, every Büchi VASS has a finite set of profiles, which we denote by $\Pi(\mathcal{V})$. Moreover, $\Pi(\mathcal{V})$ can be constructed effectively: Given a set of edges, a simple reduction to checking unboundedness of a counter can be used to check if it is a profile. Furthermore, to every run ρ of $\mathcal{V}$, we can associate a profile: The run ρ must have a corresponding run in $\mathsf{KM}(\mathcal{V})$, which has a finite set $\Pi(\rho)$ of edges that are used infinitely often. Thus, ρ decomposes as $\rho_0\rho_1$ such that ρ_1 only contains edges from π. Then, ρ_1 decomposes into $\sigma_1\sigma_2\cdots$ such that each σ_i uses every edge from $\Pi(\rho)$ at least once and starts (and ends) in a final state. Since $\leq$ is a well-quasi ordering on $\mathbb{N}^n$, there are $r < s$ such that $\delta(\sigma_r \cdots \sigma_s) \geq \mathbf{0}$. Thus, $\sigma = \sigma_r \cdots \sigma_s$ is our desired transition sequence showing that $\Pi(\rho)$ is a profile. For each $\pi \in \Pi(\mathcal{V})$, we denote by $L_\pi(\mathcal{V})$ the set of all words accepted by runs ρ of $\mathcal{V}$ for which $\Pi(\rho) = \pi$. Then clearly:

▶ **Lemma 5.2.** $L(\mathcal{V}) = \bigcup_{\pi\in\Pi(\mathcal{V})} L_\pi(\mathcal{V})$.

A system of inequalities for each profile. Our next step is to associate with each profile $\pi \in \Pi(\mathcal{V})$ a system of linear inequalities. We need some terminology. A *π-cycle* is a cycle σ in $\mathsf{KM}(\mathcal{V})$ that only contains edges in π. If in addition, σ visits each state of $\mathsf{KM}(\mathcal{V})$ at most once, except for the initial state, which is visited twice, then σ is a *primitive π-cycle*. Clearly, a primitive π-cycle has length $\leq |\pi|$. Moreover, from every π-cycle σ, one can successively cut out primitive π-cycles until it is empty. Therefore, if $\tau_1, \ldots, \tau_m$ are the primitive π-cycles of $\mathsf{KM}(\mathcal{V})$, then there are numbers $r_1, \ldots, r_m \in \mathbb{N}$ such that $\Delta(\sigma) = r_1 \cdot \Delta(\tau_1) + \cdots + r_m \cdot \Delta(\tau_m)$. We call σ a *complete π-cycle* if this holds for some $r_1, \ldots, r_m \geq 1$. Observe that if π is a profile, then this is always witnessed by a complete π-cycle: Take any cycle σ witnessing that π is a profile. Then $\sigma^{|\pi|}$ contains each primitive π-cycle as a subsequence. Hence, the cycle $\sigma^{m\cdot|\pi|}$ is complete: We can carry out the cutting in each factor $\sigma^{|\pi|}$ so as to cut some τ_i at least once. Moreover, $\sigma^{m\cdot|\pi|}$ still witnesses that π is a profile, since $\delta(\sigma^{m\cdot|\pi|}) = m \cdot |\pi| \cdot \delta(\sigma) \geq \mathbf{0}$.

Let us now construct the system of inequalities associated with π. Let σ be a complete π-cycle witnessing that π is a profile and let $\tau_1, \ldots, \tau_m$ be the primitive π-cycles. Let $\boldsymbol{A}_\pi \in \mathbb{Z}^{(m+1)\times n}$ be the matrix with rows $\varphi(\tau_1), \ldots, \varphi(\tau_m), \varphi(\sigma)$, and let $\boldsymbol{b} \in \mathbb{Z}^{m+1}$ be the column vector $(0, \ldots, 0, -1)$. Then clearly, $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ is equivalent to $\langle \boldsymbol{x}, \varphi(\sigma)\rangle < 0$ and $\langle \boldsymbol{x}, \varphi(\tau)\rangle \leq 0$ for each primitive π-cycle τ.

Inseparability flowers. An *inseparability flower* is a structure in the Karp-Miller graph $\mathsf{KM}(\mathcal{V})$ as depicted below. It consists of a final state q and three cycles α, β, γ that all start in q and that meet the given conditions.



$\delta(\alpha\beta\gamma) \geq \mathbf{0}$
$\varphi(\alpha\beta) \geq \mathbf{0}$
$\varphi(\alpha\beta\gamma) \in \mathbb{Q} \cdot \varphi(\alpha)$

Let us give some intuition on why such a flower is the relevant structure to look for. True to its name, an inseparability flower guarantees the existence of an inseparability witness, i.e. a family of words accepted by the pumpable Büchi VASS $\mathcal{V}$ that escape every basic separator $S_{\boldsymbol{x},k}$. Such a family of words therefore needs an accepting run for each member, and the three conditions of the flower provide such runs: The first condition ensures that the three cycles actually correspond to a transition sequence enabled in $\mathcal{V}$. The second condition guarantees that for every $\boldsymbol{x} \in \mathbb{N}^n$, the $\boldsymbol{x}$-weighted letter balance of α or of β is positive; unless they are both zero, in which case the third condition ensures that $\alpha\beta\gamma$ has $\boldsymbol{x}$-weighted balance zero. This allows us to construct, for each k, a run that escapes $S_{\boldsymbol{x},k}$ for all $\boldsymbol{x}$: By sufficiently

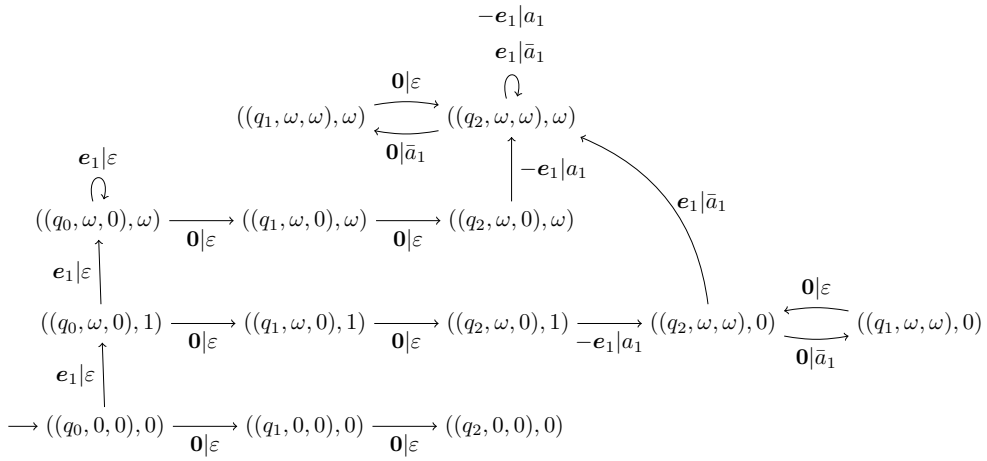


Figure 3 The Karp-Miller graph $\mathsf{KM}(\mathcal{V}_{\mathsf{pump}})$ of the Büchi VASS $\mathcal{V}_{\mathsf{pump}}$ from Figure 2(left). Here we did not mark the final states to reduce visual clutter; every state that includes q_1 is considered final. For similar reasons, we also only labelled the edges of the graph with letters and counter effects. The proper edge labels would be full transitions of $\mathcal{V}_{\mathsf{pump}}$, including source and target state.

repeating each cycle α, β, and γ, we obtain a run that for each $\boldsymbol{x} \in \mathbb{N}^n$, will either (i) have infixes with $\boldsymbol{x}$-weighted balance $> k$, or (ii) attain some $\boldsymbol{x}$-weighted balance infinitely often. Each of these properties rules out membership in $S_{\boldsymbol{x},k}$. Proposition 5.5 proves this formally.

▶ **Theorem 5.3.** *Let $\mathcal{V}$ be a Büchi VASS such that $L(\mathcal{V})$ is pumpable. Then the following are equivalent: (i) $L(\mathcal{V}) \not| D_n$. (ii) There is a profile $\pi \in \Pi(\mathcal{V})$ such that the system $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ has no solution $\boldsymbol{x} \in \mathbb{N}^n$. (iii) There exists an inseparability flower in $\mathsf{KM}(\mathcal{V})$.*

The decision procedure. Before we prove Theorem 5.3, let us see how to use it to decide separability. Given Büchi VASS $\mathcal{V}_1$ and $\mathcal{V}_2$, we can compute $\mathcal{V}$ so that $L(\mathcal{V}_1) \mid L(\mathcal{V}_2)$ if and only if $L(\mathcal{V}) \mid D_n$, by Lemma 3.4. Then Theorem 3.7 tells us that $L(\mathcal{V}_{\mathsf{pump}})$ is pumpable and we have $L(\mathcal{V}) \mid D_n$ if and only if $L(\mathcal{V}_{\mathsf{pump}}) \mid D_n$. Finally, by Theorem 5.3, we can check whether $L(\mathcal{V}_{\mathsf{pump}}) \mid D_n$ by checking the systems $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ for satisfiability: If there is a solution for every $\pi \in \Pi(\mathcal{V}_{\mathsf{pump}})$, then we have separability; otherwise, we have inseparability. Since the systems $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ are constructed directly from $\mathsf{KM}(\mathcal{V}_{\mathsf{pump}})$, we need to explicitly construct the latter. Therefore our procedure may take Ackermann time, because Karp-Miller graphs can be Ackermann large [31, Theorem 2].

▶ **Example 5.4.** Consider the instance of regular separability where our two inputs are the Büchi VASS $\mathcal{V}$ found in Figure 1(left), and another Büchi VASS accepting the language D_1. Since we are already in the case of wanting to decide $L(\mathcal{V}) \mid D_1$, we can skip the first step of applying Lemma 3.4. The second step is to apply Theorem 3.7 and construct $\mathcal{V}_{\mathsf{pump}}$, which we have already done for this case in Figure 2(right).

Now we have to construct $\mathsf{KM}(\mathcal{V}_{\mathsf{pump}})$, which can be found in Figure 3. There are two relevant parts of $\mathsf{KM}(\mathcal{V}_{\mathsf{pump}})$, where we can find cycles involving a final state: (1) the part on the right, where the state tuples contain ω twice and the counter value is 0, and (2) the part at the top with triple ωs. In the following we will only write down the states, as the counter values and the other contents of the state tuples will be clear from context.

For part (1), the Büchi VASS $\mathcal{V}_{\mathsf{pump}}$ has only a single profile π_1 containing only the two edges between q_1 and q_2. Since each π_1-cycle σ only consists of repetitions of the primitive cycle $q_1 \xrightarrow{\mathbf{0}|\varepsilon} q_2 \xrightarrow{\mathbf{0}|\bar{a}_1} q_1$, we have $\varphi(\sigma) < 0$. Therefore the system $\boldsymbol{A}_{\pi_1}\boldsymbol{x} \leq \boldsymbol{b}$ trivially has a solution $\boldsymbol{x} = 1$.

Regarding part (2), $\mathcal{V}_{\mathsf{pump}}$ has exactly two more profiles: profile π_2 containing only the two edges between q_1 and q_2, and profile π_3, which additionally contains the two loop edges on q_2. The cycles of π_2 look almost exactly like the cycles of π_1 with only the counter values of the nodes in the graph being different. Thus, the system $\boldsymbol{A}_{\pi_2}\boldsymbol{x} \leq \boldsymbol{b}$ is the exact same system as $\boldsymbol{A}_{\pi_1}\boldsymbol{x} \leq \boldsymbol{b}$ and also trivially has a solution $\boldsymbol{x} = 1$.

For π_3, we have as primitive cycles both the loop edges on q_2 as well as the primitive cycle of π_2. To obtain a complete π_3-cycle, we simply insert both loops into the π_2-cycle at q_2 forming the cycle $\sigma = q_1 \xrightarrow{\mathbf{0}|\varepsilon} q_2 \xrightarrow{-\boldsymbol{e}_1|a_1} q_2 \xrightarrow{\boldsymbol{e}_1|\bar{a}_1} q_2 \xrightarrow{\mathbf{0}|\bar{a}_1} q_1$. Since σ contains all primitive cycles exactly once without overlap, it is automatically complete. We also have $\delta(\sigma) = 0$, meaning σ is a cycle witnessing π_3 as a profile. Thus these cycles lead to the following system of inequalities $\boldsymbol{A}_{\pi_3}\boldsymbol{x} \leq \boldsymbol{b}$:

$$\begin{aligned}
1 \cdot x_1 &\leq 0 && \text{loop 1}\\
-1 \cdot x_1 &\leq 0 && \text{loop 2}\\
-1 \cdot x_1 &\leq 0 && \pi_2\text{-cycle}\\
-1 \cdot x_1 &\leq -1 && \text{complete } \pi_3\text{-cycle}
\end{aligned}$$

Clearly this system has no solution; the first and last inequality are contradictory. Therefore we conclude regular inseparability for $L(\mathcal{V})$ and D_1.

While not part of the decision procedure, for an inseparable instance of the problem as we have here, we can also find an inseparability flower in $\mathsf{KM}(\mathcal{V}_{\mathsf{pump}})$. In this case we have $\alpha = q_1 \xrightarrow{\mathbf{0}|\varepsilon} q_2 \xrightarrow{\mathbf{0}|\bar{a}_1} q_1$, $\beta = q_1 \xrightarrow{\mathbf{0}|\varepsilon} q_2 \xrightarrow{-\boldsymbol{e}_1|a_1} q_2 \xrightarrow{-\boldsymbol{e}_1|a_1} q_2 \xrightarrow{\mathbf{0}|\bar{a}_1} q_1$, and $\gamma = q_1 \xrightarrow{\mathbf{0}|\varepsilon} q_2 \xrightarrow{\boldsymbol{e}_1|\bar{a}_1} q_2 \xrightarrow{\boldsymbol{e}_1|\bar{a}_1} q_2 \xrightarrow{\mathbf{0}|\bar{a}_1} q_1$. This selection of cycles meets all the requirements of a flower: $\delta(\alpha\beta\gamma) = 0$, $\varphi(\alpha\beta) = 0$, and $\varphi(\alpha\beta\gamma) = -3 = 3 \cdot \varphi(\alpha)$.

Inseparability flowers disprove separability. The remainder of this section is devoted to proving Theorem 5.3. The implication "(i)⇒(ii)" follows by applying Corollary 4.4 to $\mathsf{KM}(\mathcal{V})$, viewed as a Büchi automaton (see full version). For "(iii)⇒(i)", we employ Lemma 3.6:

▶ **Proposition 5.5.** *If $L(\mathcal{V})$ is pumpable and $\mathsf{KM}(\mathcal{V})$ has an insep. flower, then $L(\mathcal{V}) \not| D_n$.*

Proof. Suppose there is an inseparability flower α, β, γ in $\mathsf{KM}(\mathcal{V})$ and also $L(\mathcal{V}) \mid D_n$. By Lemma 3.6, there is a $k \in \mathbb{N}$ and a finite set $X \subseteq \mathbb{N}^n$ such that $L(\mathcal{V}) \subseteq \bigcup_{\boldsymbol{x}\in X} S_{\boldsymbol{x},k}$. We claim that for every $\boldsymbol{x} \in \mathbb{N}^n$, at least one of the following holds:

$$\langle \boldsymbol{x}, \varphi(\alpha)\rangle > 0, \qquad \langle \boldsymbol{x}, \varphi(\beta)\rangle > 0, \qquad \text{or } \langle \boldsymbol{x}, \varphi(\alpha\beta\gamma)\rangle = 0. \tag{1}$$

Indeed, if $\langle \boldsymbol{x}, \varphi(\alpha)\rangle \leq 0$ and $\langle \boldsymbol{x}, \varphi(\beta)\rangle \leq 0$, then $\varphi(\alpha\beta) \geq \mathbf{0}$ implies that $\langle \boldsymbol{x}, \varphi(\alpha)\rangle = \langle \boldsymbol{x}, \varphi(\beta)\rangle = 0$. Since $\varphi(\alpha\beta\gamma) = N \cdot \varphi(\alpha)$ for some $N \in \mathbb{Q}$, we have $\varphi(\alpha\beta\gamma) = \mathbf{0}$. This proves the claim. Because of (1), the sequence $\alpha^{k+1}\beta^{k+1}\gamma^{k+1}$ either has an infix χ with $\langle \boldsymbol{x}, \varphi(\chi)\rangle > k$ or we have $\langle \boldsymbol{x}, \varphi(\alpha^{k+1}\beta^{k+1}\gamma^{k+1})\rangle = 0$. Since $\delta(\alpha^{k+1}\beta^{k+1}\gamma^{k+1}) \geq \mathbf{0}$, there is a run ρ such that $\rho\alpha^{k+1}\beta^{k+1}\gamma^{k+1}$ is a run in $\mathcal{V}$. Hence, $\rho(\alpha^{k+1}\beta^{k+1}\gamma^{k+1})^\omega$ is a run in $\mathcal{V}$ whose word cannot belong to $S_{\boldsymbol{x},k}$ for any $\boldsymbol{x} \in \mathbb{N}^n$, contradicting $L(\mathcal{V}) \subseteq \bigcup_{\boldsymbol{x}\in X} S_{\boldsymbol{x},k}$. ◀

Constructing inseparability flowers. It remains to show the implication "(ii)⇒(iii)". Suppose there is a profile $\pi \in \Pi(\mathcal{V})$ whose associated system of inequalities $\boldsymbol{A}_\pi \boldsymbol{x} \leq \boldsymbol{b}$ is unsatisfiable. By Farkas' Lemma, there exists a $\boldsymbol{y} \in \mathbb{N}^{m+1}$ such that $\boldsymbol{y}^\top \boldsymbol{A}_\pi \geq \boldsymbol{0}$ and $\boldsymbol{y}^\top \boldsymbol{b} < 0$. From this vector $\boldsymbol{y}$, we now construct an inseparability flower in $\mathsf{KM}(\mathcal{V})$.

Let σ be the complete π-cycle in $\mathsf{KM}(\mathcal{V})$ that was chosen to construct $\boldsymbol{A}_\pi$. Let $\tau_1, \ldots, \tau_m$ be the primitive π-cycles. Since σ is complete, there is a vector $\boldsymbol{r} = (r_1, \ldots, r_m) \in \mathbb{N}^m$ so that $r_1, \ldots, r_m \geq 1$ and $\Delta(\sigma) = r_1 \cdot \Delta(\tau_1) + \cdots + r_m \cdot \Delta(\tau_m)$. Moreover, since σ contains every edge of π, we can wlog. write $\sigma = \sigma_0 \cdots \sigma_m$ such that between σ_{i-1} and σ_i, σ arrives in the initial state of τ_i. The decomposition allows us to insert further repetitions of the primitive cycles. For $\boldsymbol{z} = (z_1, \ldots, z_m) \in \mathbb{N}^m$ with $\boldsymbol{z} \geq \boldsymbol{r}$, we define $\sigma^{\boldsymbol{z}}$ as $\sigma_0 \tau_1^{z_1 - r_1} \sigma_1 \cdots \tau_m^{z_m - r_m} \sigma_m$. Then $\Delta(\sigma^{\boldsymbol{z}}) = z_1 \cdot \Delta(\tau_1) + \cdots + z_m \cdot \Delta(\tau_m)$. In particular, for $\boldsymbol{s}, \boldsymbol{t} \geq \boldsymbol{r}$, we have $\Delta(\sigma^{\boldsymbol{s}} \sigma^{\boldsymbol{t}}) = \Delta(\sigma^{\boldsymbol{s}+\boldsymbol{t}})$.

Recall that every transition in a Karp-Miller graph is labeled by a VASS transition, and so every transition sequence χ in $\mathsf{KM}(\mathcal{V})$ is labeled by a transition sequence in $\mathcal{V}$, which we denote by $\mathsf{trans}(\chi)$. We now define the transition sequences α, β, and γ as $\mathsf{trans}(\sigma^{\boldsymbol{z}})$ for suitable vectors $\boldsymbol{z}$. For α, we take $\mathsf{trans}(\sigma)$, the transitions labeling the complete π-cycle. Observe that $\sigma = \sigma^{\boldsymbol{r}}$. We proceed to define $\beta = \mathsf{trans}(\sigma^{\boldsymbol{s}})$ and $\gamma = \mathsf{trans}(\sigma^{\boldsymbol{t}})$. The choice of the vectors $\boldsymbol{s}$ and $\boldsymbol{t}$ has to meet the requirements on an inseparability flower: $\varphi(\alpha\beta) \geq \boldsymbol{0}$, $\delta(\alpha\beta\gamma) \geq \boldsymbol{0}$, and $\varphi(\alpha\beta\gamma) \in \mathbb{Q} \cdot \varphi(\alpha)$.

Step I: Building β. We will define $\boldsymbol{s}$ so that $\varphi(\alpha\beta) = \varphi(\sigma^{\boldsymbol{r}}\sigma^{\boldsymbol{s}}) = \varphi(\sigma^{\boldsymbol{r}+\boldsymbol{s}}) \geq \boldsymbol{0}$. The remaining two requirements (i.e. $\delta(\alpha\beta\gamma) \geq \boldsymbol{0}$ and $\varphi(\alpha\beta\gamma) \in \mathbb{Q} \cdot \varphi(\alpha)$) will be ensured with an appropriate choice of $\boldsymbol{t}$ in Step II. Let us now describe how to pick $\boldsymbol{s}$. Recall that $\boldsymbol{y}$ is the vector from the application of Farkas' Lemma. It can be understood as assigning a repetition count y_i to every primitive cycle τ_i in the profile and a repetition count y_{m+1} to the complete π-cycle σ. Since $\boldsymbol{y}^\top \boldsymbol{A}_\pi \geq \boldsymbol{0}$, and since our goal is to make $\varphi(\alpha\beta)$ non-negative, we will use $\boldsymbol{y}$ to construct a vector $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m) \in \mathbb{N}^m$ so that $\varphi(\sigma^{\hat{\boldsymbol{y}}}) = \boldsymbol{y}^\top \boldsymbol{A}_\pi$. The right definition is $\hat{y}_i := y_i + y_{m+1} \cdot r_i$ for $i \in [1, m]$, because

$$\boldsymbol{y}^\top \boldsymbol{A}_\pi = \sum_{i=1}^{m} y_i \cdot \varphi(\tau_i) + y_{m+1} \cdot \varphi(\sigma) = \sum_{i=1}^{m} (y_i + y_{m+1} r_i) \varphi(\tau_i) = \varphi(\sigma^{\hat{\boldsymbol{y}}}).$$

We now choose $M \in \mathbb{N}$ such that $\boldsymbol{s} = M \cdot \hat{\boldsymbol{y}} - \boldsymbol{r} \geq \boldsymbol{r}$. This is possible since all entries in $\hat{\boldsymbol{y}}$ are positive, due to $y_{m+1} > 0$ by $\boldsymbol{y}^\top \boldsymbol{b} < 0$, and $r_i > 0$ for all i by definition. Then we have $\varphi(\alpha\beta) = \varphi(\sigma^{\boldsymbol{r}}\sigma^{\boldsymbol{s}}) = \varphi(\sigma^{\boldsymbol{r}+\boldsymbol{s}}) = \varphi(\sigma^{M \cdot \hat{\boldsymbol{y}}}) = M \cdot \varphi(\sigma^{\hat{\boldsymbol{y}}}) \geq \boldsymbol{0}$.

Step II: Building γ. It remains to define $\boldsymbol{t}$ so that $\gamma = \mathsf{trans}(\sigma^{\boldsymbol{t}})$ satisfies $\delta(\alpha\beta\gamma) = \delta(\sigma^{\boldsymbol{r}+\boldsymbol{s}+\boldsymbol{t}}) \geq \boldsymbol{0}$ and $\varphi(\alpha\beta\gamma) \in \mathbb{Q} \cdot \varphi(\alpha)$. The idea is to choose $\boldsymbol{t}$ so that $\boldsymbol{r} + \boldsymbol{s} + \boldsymbol{t}$ is a positive multiple of $\boldsymbol{r}$. Such a choice is possible, because $\boldsymbol{r}$ has positive entries everywhere: We pick $N \in \mathbb{N}$ such that $\boldsymbol{t} := N \cdot \boldsymbol{r} - \boldsymbol{s} - \boldsymbol{r} \geq \boldsymbol{r}$. Then indeed $\delta(\alpha\beta\gamma) = \delta(\sigma^{\boldsymbol{r}+\boldsymbol{s}+\boldsymbol{t}}) = \delta(\sigma^{N \cdot \boldsymbol{r}}) = N \cdot \delta(\sigma^{\boldsymbol{r}}) = N \cdot \delta(\sigma) \geq \boldsymbol{0}$ and $\varphi(\alpha\beta\gamma) = \varphi(\sigma^{\boldsymbol{r}+\boldsymbol{s}+\boldsymbol{t}}) = \varphi(\sigma^{N \cdot \boldsymbol{r}}) = N \cdot \varphi(\sigma^{\boldsymbol{r}}) = N \cdot \varphi(\alpha)$.

6 One-dimensional Büchi VASS

Our second contribution is the precise complexity of separability for the 1-dimensional case.

▶ **Theorem 6.1.** *Regular separability for 1-dimensional Büchi VASS with binary encoded updates is* PSPACE*-complete.*

For the lower bound, we use a simple reduction from the disjointness problem $L_1 \cap L_2 \stackrel{?}{=} \emptyset$ for finite-word languages of 1-dim. VASS [20]. However, we can also show that separability is PSPACE-hard even if the input languages are promised to be disjoint.

For the upper bound, we rely on the results in Section 5, but need a modification. There, to simplify the exposition, we first make the input language pumpable, which may incur an Ackermannian blowup. A closer look at the results, however, reveals that we can also check separability directly on the Karp-Miller graph of $\bar{\mathcal{V}}$ as defined in Section 3.

▶ **Proposition 6.2.** *Let $\mathcal{V}$ be a Büchi VASS with $L(\mathcal{V}) \subseteq \Sigma_n^\omega$. Then $L(\mathcal{V}) \not| D_n$ if and only if $\mathsf{KM}(\bar{\mathcal{V}})$ has an inseparability flower.*

Proposition 6.2 allows us to phrase inseparability as the existence of a run in $\bar{\mathcal{V}}$ that satisfies certain constraints. Recall that if $\mathcal{V}$ is 1-dimensional and over Σ_1, then $\bar{\mathcal{V}}$ has two counters the second of which tracks the letter balance.

▶ **Corollary 6.3.** *Let $\mathcal{V}$ be a 1-dimensional Büchi VASS with $L(\mathcal{V}) \subseteq \Sigma_1^\omega$ and $L(\mathcal{V}) \cap D_1 = \emptyset$. Then $L(\mathcal{V}) \not| D_1$ if and only if there exist states p, q, r with r final, and a run in $\bar{\mathcal{V}}$ as follows:*

$$(q_0,0,0) \xrightarrow{*} \overbrace{(p,x_1,y_1) \xrightarrow{*} (p,x_2,y_2)}^{\sigma_1} \xrightarrow{*} \overbrace{(q,x_3,y_3) \xrightarrow{*} (q,x_4,y_4)}^{\sigma_2}$$
$$\xrightarrow{*} \overbrace{(r,x_5,y_5) \xrightarrow{*} \underbrace{(r,x_6,y_6)}_{\beta} \xrightarrow{*} (r,x_7,y_7)}^{\alpha} \xrightarrow{*} \overbrace{(r,x_8,y_8)}^{\gamma}$$

(1) $y_3 < y_4$ *and also (a)* $x_3 \leq x_4$ *or (b)* $x_1 < x_2$ *and* $y_1 \leq y_2$
(2) $y_5 \leq y_7$
(3) $x_5 \leq x_8$
(4) *if* $y_5 = y_6$, *then* $y_5 = y_8$.

Observe that an inseparability flower in $\mathsf{KM}(\bar{\mathcal{V}})$ must carry ω in the second coordinate, meaning the letter balance is unbounded. Otherwise, it would yield an accepting run of $\bar{\mathcal{V}}$, which cannot exist because $L(\mathcal{V}) \cap D_1 = \emptyset$. If the flower has ω in the second coordinate, we can construct a *finite* run as above. The cycles σ_1 and σ_2 plus Condition 1 ensure that indeed the second coordinate becomes ω. Condition 2 is $\varphi(\alpha\beta) \geq \mathbf{0}$. Condition 3 says $\delta(\alpha\beta\gamma) \geq \mathbf{0}$. Finally, to express $\varphi(\alpha\beta\gamma) \in \mathbb{Q} \cdot \varphi(\alpha)$, note that for integers $a \in \mathbb{Q} \cdot b$ iff $b = 0$ implies $a = 0$. Condition 4 expresses that $y_6 - y_5 = 0$ implies $y_8 - y_5 = 0$.

In order to apply Corollary 6.3 for deciding $L(\mathcal{V}_1) \mid L(\mathcal{V}_2)$ for 1-dim. Büchi VASS $\mathcal{V}_1, \mathcal{V}_2$ with binary counter updates, we would like to follow the approach for the general case and use Lemma 3.4 to first construct $\mathcal{V}$ so that $L(\mathcal{V}_1) \mid L(\mathcal{V}_2)$ if and only if $L(\mathcal{V}) \mid D_1$. From $\mathcal{V}$, we would then construct the 2-dimensional Büchi VASS $\bar{\mathcal{V}}$ that tracks the letter balance, and on $\bar{\mathcal{V}}$ we would then check the conditions of Corollary 6.3. The problem is that, under binary updates, the intermediary $\mathcal{V}$ may become exponentially large. We use the fact that also $\bar{\mathcal{V}}$ has binary counters available. This allows us to directly construct a compact variant of $\bar{\mathcal{V}}$:

▶ **Lemma 6.4.** *Given 1-dim. Büchi VASS $\mathcal{V}_1, \mathcal{V}_2$ with binary updates, there is a a 1-dim. Büchi VASS $\mathcal{V}$ with $L(\mathcal{V}_1) \cap L(\mathcal{V}_2) = \emptyset$ iff $L(\mathcal{V}) \cap D_1 = \emptyset$, $L(\mathcal{V}_1) \mid L(\mathcal{V}_2)$ iff $L(\mathcal{V}) \mid D_1$, and we can construct in time polynomial in $|\mathcal{V}_1| + |\mathcal{V}_2|$ the 2-dim. Büchi VASS $\bar{\mathcal{V}}$ (binary updates).*

Detecting constrained runs in 2-VASS. It remains to check for the existence of runs in $\bar{\mathcal{V}}$ as described in Corollary 6.3, and to check whether $L(\mathcal{V}_1) \cap L(\mathcal{V}_2) = \emptyset$. Both of these problems reduce to what we call the *constrained runs problem for 2-VASS*. Recall that *Presburger arithmetic* is the first-order theory of $(\mathbb{N}, +, <, 0, 1)$. We will use the existential fragment to express conditions on counter values of VASS like the ones from Corollary 6.3. The *constrained runs problem* is the following:

Given A 2-dim. VASS $\mathcal{V}$ (with updates encoded in binary), a number $m \in \mathbb{N}$, states $q_1, \ldots, q_m$ in $\mathcal{V}$, a quantifier-free Presburger formula $\psi(x_1, y_1, \ldots, x_m, y_m)$, and $s, t \in [1, m]$, $s \leq t$.
Question Does there exist a run $(q_0, 0, 0) \xrightarrow{*} (q_1, x_1, y_1) \xrightarrow{*} \cdots \xrightarrow{*} (q_m, x_m, y_m)$ that visits a final state between (q_s, x_s, y_s) and (q_t, x_t, y_t) and satisfies $\psi(x_1, y_1, \ldots, x_m, y_m)$?

Lemma 6.4 and Corollary 6.3 imply that if $L(\mathcal{V}_1) \cap L(\mathcal{V}_2) = \emptyset$, then $L(\mathcal{V}_1) \mid L(\mathcal{V}_2)$ reduces to the constrained runs problem on $\bar{\mathcal{V}}$. Moreover, checking $L(\mathcal{V}_1) \cap L(\mathcal{V}_2) = \emptyset$ reduces via a product construction to checking emptiness of a 2-VASS. Such a 2-VASS has an accepting run iff $(q_0, 0, 0) \xrightarrow{*} (q, x, y) \xrightarrow{*} (q, x', y')$ with $(x, y) \leq (x', y')$ and q final. Hence, this problem also reduces to the constrained runs problem for 2-VASS. We thus need to show:

▶ **Proposition 6.5.** *The constrained runs problem for* 2*-VASS is solvable in* PSPACE*.*

For Proposition 6.5, we show that if there is a constrained run, then there is one with at most exponential counter values along the way. For this, we use methods from [4].

Complexity in higher dimension. We leave open two natural questions: (i) What is the complexity of regular separability for Büchi d-VASS, for each $d \geq 2$? (ii) What is the complexity of regular separability for Büchi VASS (where the dimension is part of the input)?

Given that the regular separability and the disjointness problem usually (but not always [27, 42]) coincide regarding decidability, we expect the complexity of regular separability to be PSPACE in every fixed dimension d and EXPSPACE in general. The lower bounds follow from Theorem 6.1 for fixed d and from [13] (because disjointness is EXPSPACE-complete [19, 29]). However, it is not clear how to show the upper bounds.

The clearest obstacle is that inseparability flowers involve a non-linear condition: The requirement $\varphi(\alpha\beta\gamma) \in \mathbb{Q} \cdot \varphi(\alpha)$ is not expressible in Presburger arithmetic. There are several generic results providing EXPSPACE upper bounds for detecting particular types of runs in VASS [17, 2, 3]. However, the numerical properties directly expressible there are confined to Presburger arithmetic. The only reason we could obtain the PSPACE upper bound for $d = 1$ is that the non-linear condition degenerates into a linear condition in dimension one: It is equivalent to "$\varphi(\alpha\beta\gamma) = 0$ or $\varphi(\alpha) \neq 0$".

References

1 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Vrunda Dave, and Shankara Narayanan Krishna. On the Separability Problem of String Constraints. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 16:1–16:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.16.

2 Mohamed Faouzi Atig and Peter Habermehl. On Yen's Path Logic for Petri Nets. *Int. J. Found. Comput. Sci.*, 22(4):783–799, 2011. doi:10.1142/S0129054111008428.

3 Michel Blockelet and Sylvain Schmitz. Model checking coverability graphs of vector addition systems. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 – 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2011. doi:10.1007/978-3-642-22993-0_13.

4 Michael Blondin, Matthias Englert, Alain Finkel, Stefan Göller, Christoph Haase, Ranko Lazic, Pierre McKenzie, and Patrick Totzke. The Reachability Problem for Two-Dimensional Vector Addition Systems with States. *J. ACM*, 68(5):34:1–34:43, 2021. doi:10.1145/3464794.

5 Heino Carstensen. Infinite behaviour if deterministic petri nets. In Michal Chytil, Ladislav Janiga, and Václav Koubek, editors, *Mathematical Foundations of Computer Science 1988, MFCS'88, Carlsbad, Czechoslovakia, August 29 – September 2, 1988, Proceedings*, volume 324 of *Lecture Notes in Computer Science*, pages 210–219. Springer, 1988. doi:10.1007/BFb0017144.

6 Pierre Chambart, Alain Finkel, and Sylvain Schmitz. Forward analysis and model checking for trace bounded WSTS. *Theor. Comput. Sci.*, 637:1–29, 2016. doi:10.1016/j.tcs.2016.04.020.

7 Christian Choffrut, Flavio D'Alessandro, and Stefano Varricchio. On the separability of sparse context-free languages and of bounded rational relations. *Theor. Comput. Sci.*, 381(1-3):274–279, 2007. doi:10.1016/j.tcs.2007.04.003.

8 Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Regular Separability of Parikh Automata. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 117:1–117:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.117.

9 Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Separability of Reachability Sets of Vector Addition Systems. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 24:1–24:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.24.

10 Lorenzo Clemente, Slawomir Lasota, and Radoslaw Piórkowski. Timed Games and Deterministic Separability. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 121:1–121:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.121.

11 Wojciech Czerwiński, Piotr Hofman, and Georg Zetzsche. Unboundedness problems for languages of vector addition systems. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *Proc. of the 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 119:1–119:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2018.119.

12 Wojciech Czerwinski and Slawomir Lasota. Regular separability of one counter automata. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005079.

13 Wojciech Czerwinski, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory (CONCUR 2018)*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CONCUR.2018.35.

14 Wojciech Czerwinski, Wim Martens, and Tomás Masopust. Efficient Separability of Regular Languages by Subsequences and Suffixes. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2013. doi:10.1007/978-3-642-39212-2_16.

15 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, Marc Zeitoun, and Georg Zetzsche. A Characterization for Decidable Separability by Piecewise Testable Languages. *Discrete Mathematics and Theoretical Computer Science*, 19(4), 2017. doi:10.23638/DMTCS-19-4-1.

16 Wojciech Czerwiński and Georg Zetzsche. An Approach to Regular Separability in Vector Addition Systems. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *Proc. of the Thirty-Fifth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2020)*, pages 341–354. ACM, 2020. doi:10.1145/3373718.3394776.

17 Stéphane Demri. On selective unboundedness of VASS. *J. Comput. Syst. Sci.*, 79(5):689–713, 2013. doi:10.1016/j.jcss.2013.01.014.

18 Jacques Duparc, Olivier Finkel, and Jean-Pierre Ressayre. The wadge hierarchy of petri nets ω-languages. In Vasco Brattka, Hannes Diener, and Dieter Spreen, editors, *Logic, Computation, Hierarchies*, volume 4 of *Ontos Mathematical Logic*, pages 109–138. De Gruyter, 2014. doi:10.1515/9781614518044.109.

19 Javier Esparza. Decidability and complexity of Petri net problems – an introduction. In G. Rozenberg and W. Reisig, editors, *Lectures on Petri Nets I: Basic Models. Advances in Petri Nets*, number 1491 in Lecture Notes in Computer Science, pages 374–428, 1998.

20 John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015. doi:10.1016/j.ic.2014.12.004.

21 Olivier Finkel. Borel ranks and wadge degrees of context free omega-languages. *Math. Struct. Comput. Sci.*, 16(5):813–840, 2006. doi:10.1017/S0960129506005597.

22 Olivier Finkel and Michal Skrzypczak. On the expressive power of non-deterministic and unambiguous petri nets over infinite words. *Fundam. Informaticae*, 183(3-4):243–291, 2021. doi:10.3233/FI-2021-2088.

23 Peter Habermehl. On the Complexity of the Linear-Time μ-calculus for Petri-Nets. In *ICATPN*, volume 1248 of *LNCS*, pages 102–116. Springer, 1997.

24 Christopher Hugenroth. Separating Regular Languages over Infinite Words with Respect to the Wagner Hierarchy. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 46:1–46:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.46.

25 Harry B. Hunt III. On the Decidability of Grammar Problems. *Journal of the ACM*, 29(2):429–447, 1982. doi:10.1145/322307.322317.

26 Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969. doi:10.1016/S0022-0000(69)80011-5.

27 Eryk Kopczynski. Invisible Pushdown Languages. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 867–872. ACM, 2016. doi:10.1145/2933575.2933579.

28 Jérôme Leroux and Sylvain Schmitz. Demystifying Reachability in Vector Addition Systems. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 56–67. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.16.

29 Richard Lipton. The reachability problem is exponential-space hard. *Yale University, Department of Computer Science, Report*, 62, 1976.

30 Tomás Masopust. Separability by piecewise testable languages is PTime-complete. *Theor. Comput. Sci.*, 711:109–114, 2018. doi:10.1016/j.tcs.2017.11.004.

31 Ernst W Mayr and Albert R Meyer. The complexity of the finite containment problem for Petri nets. *Journal of the ACM (JACM)*, 28(3):561–576, 1981.

32 Théo Pierron, Thomas Place, and Marc Zeitoun. Quantifier Alternation for Infinite Words. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures – 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 2016. doi:10.1007/978-3-662-49630-5_14.

33 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating Regular Languages by Locally Testable and Locally Threshold Testable Languages. In Anil Seth and Nisheeth K. Vishnoi, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, volume 24 of *LIPIcs*, pages 363–375. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.FSTTCS.2013.363.

34 Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14–18, 2014*, pages 75:1–75:10. ACM, 2014. doi:10.1145/2603088.2603098.

35 Thomas Place and Marc Zeitoun. Separation and the Successor Relation. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 662–675. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.662.

36 Thomas Place and Marc Zeitoun. Separating Regular Languages with First-Order Logic. *Log. Methods Comput. Sci.*, 12(1), 2016. doi:10.2168/LMCS-12(1:5)2016.

37 Thomas Place and Marc Zeitoun. Separating Without Any Ambiguity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 137:1–137:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.137.

38 Thomas Place and Marc Zeitoun. Separation and covering for group based concatenation hierarchies. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785655.

39 Andreas Podelski and Andrey Rybalchenko. A Complete Method for the Synthesis of Linear Ranking Functions. In *VMCAI*, volume 2937 of *LNCS*, pages 239–251. Springer, 2004.

40 Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

41 Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2), 1976.

42 Ramanathan S. Thinniyam and Georg Zetzsche. Regular Separability and Intersection Emptiness are Independent Problems. In *Proc. of the 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019)*, volume 150 of *LIPIcs*, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

43 Rüdiger Valk. Infinite behaviour of petri nets. *Theoretical computer science*, 25(3):311–341, 1983.

44 Georg Zetzsche. Separability by piecewise testable languages and downward closures beyond subwords. In Anuj Dawar and Erich Grädel, editors, *Proc. of the Thirty-Third Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2018)*, pages 929–938. ACM, 2018. doi:10.1145/3209108.3209201.