On the Context-Freeness Problem for Vector Addition Systems

Jérôme Leroux, Vincent Penelle, Grégoire Sutre

On the Context-Freeness Problem for Vector Addition Systems

Jérôme Leroux
LaBRI, UMR CNRS 5800
University of Bordeaux
Talence, France

Vincent Penelle
LIGM, UMR CNRS 8049
University of Paris Est
Marne-la-Vallée, France

Grégoire Sutre
LaBRI, UMR CNRS 5800
University of Bordeaux
Talence, France

***Abstract*—Petri nets, or equivalently vector addition systems (VAS), are widely recognized as a central model for concurrent systems. Many interesting properties are decidable for this class, such as boundedness, reachability, regularity, as well as context-freeness, which is the focus of this paper. The context-freeness problem asks whether the trace language of a given VAS is context-free. This problem was shown to be decidable by Schwer in 1992, but the proof is very complex and intricate. The resulting decision procedure relies on five technical conditions over a customized coverability graph. These five conditions are shown to be necessary, but the proof that they are sufficient is only sketched. In this paper, we revisit the context-freeness problem for VAS, and give a simpler proof of decidability. Our approach is based on witnesses of non-context-freeness, that are bounded regular languages satisfying a nesting condition. As a corollary, we obtain that the trace language of a VAS is context-free if, and only if, it has a context-free intersection with every bounded regular language.**

***Keywords*-Vector addition systems, Petri nets, context-freeness, pushdown automata, bounded languages, semilinear sets.**

I. Introduction

Petri nets, or equivalently *vectors addition systems* (VAS), are arguably one of the most studied formalisms for the modeling and analysis of concurrent systems. Despite their fairly large expressive power, many verification problems are decidable for VAS: boundedness, reachability, liveness, regularity, etc. [1]. A large subset of these decidable problems can be solved using the Karp & Miller coverability graph [2]. For instance, regularity can be solved by checking that no cycle of the coverability graph has a negative component [3]. The coverability graph may have an Ackermannian size [4], but most verification problems that are known to be solvable using the coverability graph can also be decided in exponential space, using Rackoff's technique [5], [6], [7].

From a language viewpoint, VAS languages contain regular languages as well as bounded context-free languages, but they are incomparable with context-free languages [8]. The characterization of the VAS languages that are context-free was left open in [8]. In this paper, we focus on *trace languages* of VAS, that is, without action labeling nor acceptance condition. Schwer showed in [9] that context-freeness is decidable for trace languages of VAS. However, the proof is long (almost 50 pages), very complex and intricate, and the resulting decision procedure is based on five technical conditions on an unfolding of the coverability graph that ensures the "iterability" of loops (see also [10]). These five conditions are shown to be necessary, but the proof that they are sufficient is only sketched.

Contribution. This paper revisits the context-freeness problem for trace languages of VAS (hereafter called the context-freeness problem for VAS). Our approach is based on a characterization by Ginsburg of bounded context-free languages [11]. Recall that a language L is called *bounded* when $L \subseteq \sigma_1^* \cdots \sigma_k^*$ for some words $\sigma_1, \ldots, \sigma_k$. We introduce a decidable sufficient condition, called *witness of non-context-freeness*, that guarantees that the trace language of a VAS intersected with $\sigma_1^* \cdots \sigma_k^*$ is not context-free.

Conversely, to detect that the trace language of a VAS is context-free, we simulate its behavior by an extended pushdown automaton, called a *vector pushdown automaton*. This automaton is parameterized by a finite language W that is used to restrict its state space. Two sink states $\bot$ and $\lightning$ are introduced to model failures of the simulation. When these states are not reachable, the simulation succeeds and a (standard) pushdown automaton recognizing the trace language of the VAS is effectively computable. When the simulation fails and reaches the state $\lightning$, we extract a witness of non-context-freeness from a run reaching this failure state (thus, proving that the trace language is not context-free). The state $\bot$ is reachable only when W is not large enough, in which case we refine our simulation with another W. We show that there exists a finite language W such that $\bot$ is not reachable, thus ensuring termination of the procedure.

As main contribution of the paper, we obtain a simple and complete proof that the context-freeness problem for VAS is decidable. Compared to [9], our approach does not use Ogden's Lemma, but is solely based on Ginsburg's characterization of bounded context-free languages [11]. This allows us to work with vectors of numbers instead of words. Moreover, as a corollary, we obtain that the trace language of a VAS is context-free if, and only if, it has a context-free intersection with every bounded regular language.

Related Work. The main source of inspiration for this work is Schwer's article [9], where it is shown that the context-freeness problem for VAS is decidable. The complexity of this problem is still open. Close to our work is also the regularity

problem for VAS, that asks whether the trace language of a given VAS is regular. This problem was shown to be decidable in [3], [12]. Recently, an exponential space upper-bound was established for this problem in [13], [7]. This upper bound was obtained through witnesses of non-regularity of the form $u_1\sigma_1^* \cdots u_k\sigma_k^*$. Our research follows a similar approach, but for the context-freeness problem.

The class of bounded languages provides a powerful tool for under-approximating complex languages. In [14], it is proved that from any context-free language L, we can effectively compute a bounded context-free language $L' \subseteq L$ with the same Parikh image. More recently, in [15], bounded languages are shown to be central in many verification techniques.

The expressive power of VAS greatly increases when actions are labeled. For labeled VAS, the trace language is the image under a morphism (derived from the labeling function) of the unlabeled trace language. The regularity problem for labeled VAS was shown to be undecidable in [16]. Here, we address the context-freeness problem, but only for unlabeled VAS.

Outline. The paper is organized as follows. Section II introduces the context-freeness problem for vector addition systems. We recall Ginsburg's characterization of bounded context-free languages in Section III. Witnesses of non-context-freeness are the subject of Sections IV and V. Vector pushdown automata are introduced and studied in Section VI. Then, we show in Section VII how to simulate a VAS with a vector pushdown automaton. Section VIII derives, from the previous sections, an algorithm solving the context-freeness problem for VAS. We conclude the paper in Section IX.

II. Vector Addition Systems

We let $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$ denote the usual sets of *natural numbers*, *integers*, and *rational numbers*, respectively. For every $\mathbb{X} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}\}$ and $\# \in \{<, \leq, \geq, >\}$, we write $\mathbb{X}_{\#0} = \{x \in \mathbb{X} \mid x \# 0\}$. Vectors (of rational numbers) and sets of vectors are typeset in bold face. The ith *component* of a vector $\boldsymbol{v}$ is written $\boldsymbol{v}(i)$. We let $\boldsymbol{e}_i$ denote the ith *unit vector*, defined by $\boldsymbol{e}_i(i) = 1$ and $\boldsymbol{e}_i(j) = 0$ for every index $j \neq i$. Given a vector $\boldsymbol{v}$, we write $\|\boldsymbol{v}\|^+$ and $\|\boldsymbol{v}\|^-$ for the sets of indexes i such that $\boldsymbol{v}(i) > 0$ and $\boldsymbol{v}(i) < 0$, respectively. The *displacement* of a word $\sigma = \boldsymbol{v}_1 \cdots \boldsymbol{v}_k$ of vectors $\boldsymbol{v}_j \in \mathbb{Q}^d$ is the sum $\sum_{j=1}^{k} \boldsymbol{v}_j$, denoted by $\Delta(\sigma)$.

We now recall the main concepts of vector addition systems. Consider a *dimension* $d \in \mathbb{N}$, with $d > 0$. A *configuration* is a vector $\boldsymbol{c} \in \mathbb{N}^d$, and an *action* is a vector $\boldsymbol{a} \in \mathbb{Z}^d$. Informally, a vector addition system moves from one configuration to the next by adding an action. In particular, an action $\boldsymbol{a}$ is enabled in a configuration $\boldsymbol{c}$ if, and only if, $\boldsymbol{c} + \boldsymbol{a} \geq \boldsymbol{0}$. This operational semantics is formalized by the labeled transition relation $\rightarrow \subseteq \mathbb{N}^d \times \mathbb{Z}^d \times \mathbb{N}^d$ defined by $\boldsymbol{c} \xrightarrow{\boldsymbol{a}} \boldsymbol{c'}$ if $\boldsymbol{c'} = \boldsymbol{c} + \boldsymbol{a}$. A *run* is a finite, alternating sequence $(\boldsymbol{c}_0, \boldsymbol{a}_1, \boldsymbol{c}_1, \ldots, \boldsymbol{a}_n, \boldsymbol{c}_n)$ of configurations and actions, satisfying $\boldsymbol{c}_{i-1} \xrightarrow{\boldsymbol{a}_i} \boldsymbol{c}_i$ for all i. The word $\boldsymbol{a}_1 \cdots \boldsymbol{a}_n$ is called the *label*[1] of the run. A *trace* from a configuration $\boldsymbol{c}$ is the label of some run that starts with $\boldsymbol{c}$. Given an *initial configuration* $\boldsymbol{c}_{\text{init}} \in \mathbb{N}^d$, we let $\mathcal{T}(\boldsymbol{c}_{\text{init}})$ denote the set of all traces from $\boldsymbol{c}_{\text{init}}$.

Remark II.1. A word $\boldsymbol{a}_1 \cdots \boldsymbol{a}_n \in (\mathbb{Z}^d)^*$ is a trace from a configuration $\boldsymbol{c} \in \mathbb{N}^d$ if, and only if, it holds that $\boldsymbol{c} + \sum_{i=1}^{h} \boldsymbol{a}_i \geq \boldsymbol{0}$ for every $h \in \{1, \ldots, n\}$.

A *vector addition system*, shortly called VAS, is a pair $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ where $\boldsymbol{A}$ is a finite subset of $\mathbb{Z}^d$ and $\boldsymbol{c}_{\text{init}} \in \mathbb{N}^d$ is an initial configuration. Its operational semantics is obtained by restricting the labeled transition relation $\rightarrow$ to actions in $\boldsymbol{A}$. Accordingly, a *trace* of a VAS $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ is a trace from $\boldsymbol{c}_{\text{init}}$ that is contained in $\boldsymbol{A}^*$. The set of all traces of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$, written $\mathcal{T}(\boldsymbol{A}, \boldsymbol{c}_{\text{init}}) = \mathcal{T}(\boldsymbol{c}_{\text{init}}) \cap \boldsymbol{A}^*$, is called the *trace language* of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$. The *context-freeness problem* for VAS asks whether the trace language of a given VAS is context-free.

This problem was shown to be decidable by Schwer in [9], but the proof is very complex and intricate. In this paper, we revisit the context-freeness problem for VAS. Our main contribution is a new, simpler proof of the following theorem.

Theorem II.2 ([9])**.** *The context-freeness problem for VAS is decidable.*

First, we recall a characterization by Ginsburg of bounded context-free languages [11].

III. Bounded Context-Free Languages

In this section, we consider words over a given finite alphabet Σ. A language $L \subseteq \Sigma^*$ is *bounded* if $L \subseteq \sigma_1^* \cdots \sigma_k^*$ for some words $\sigma_1, \ldots, \sigma_k$ in Σ^*. In his book [11], Ginsburg characterizes those bounded languages that are context-free, in terms of semilinear sets. Our analysis of the context-freeness problem for VAS builds upon this characterization, so we briefly present it in this section. The reader is referred to [11] for further details.

First, we recall the notion of semilinear sets. A *periodic* set is a subset $\boldsymbol{P}$ of $\mathbb{N}^k$ such that $\boldsymbol{0} \in \boldsymbol{P}$ and $\boldsymbol{P} + \boldsymbol{P} \subseteq \boldsymbol{P}$. Given a finite set $\boldsymbol{G} \subseteq \mathbb{N}^k$ of *generators*, the periodic set *generated* by $\boldsymbol{G}$ is the least periodic set containing $\boldsymbol{G}$. Put differently, the periodic set generated by $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n\}$ is equal to $\mathbb{N}\boldsymbol{g}_1 + \cdots + \mathbb{N}\boldsymbol{g}_n$. A *linear set* is a subset of $\mathbb{N}^k$ of the form $\boldsymbol{b} + \boldsymbol{P}$, where $\boldsymbol{b} \in \mathbb{N}^k$ and $\boldsymbol{P} \subseteq \mathbb{N}^k$ is a finitely-generated periodic set. A *semilinear set* is a finite union of linear sets. Let us recall that semilinear sets coincide with the sets definable in $\mathsf{FO}(\mathbb{N}, +)$, also known as Presburger arithmetic [17].

Ginsburg's characterization of bounded context-free languages is expressed by a "stratification" requirement on the generators of periodic sets. A binary relation R on $\{1, \ldots, k\}$ is called *nested* if it satisfies the two following conditions:

$$(s,t) \in R \quad \Rightarrow \quad s \leq t \tag{1}$$

$$(r,t) \in R \wedge (s,u) \in R \quad \Rightarrow \quad \neg(r < s < t < u) \tag{2}$$

Definition III.1. *A finite set $\boldsymbol{G} \subseteq \mathbb{N}^k$ is* stratified *if there exists a nested relation R on $\{1, \ldots, k\}$ such that $\boldsymbol{G} \subseteq \bigcup_{(s,t) \in R} \mathbb{N}\boldsymbol{e}_s + \mathbb{N}\boldsymbol{e}_t$.*

[1] Observe that we do not label actions by letters of some alphabet. This corresponds to the notion of *free labeling* in Petri net terminology.

We call a finitely-generated periodic set $\boldsymbol{P}$ *stratifiable* when it is generated by a finite stratified set. By extension, a linear set $\boldsymbol{b}+\boldsymbol{P}$ is *stratifiable* if $\boldsymbol{P}$ is stratifiable[2]. We are now ready to present Ginsburg's characterization of bounded context-free languages.

Theorem III.2 ([11, p. 162])**.** *Consider a language $L \subseteq \sigma_1^* \cdots \sigma_k^*$, where each $\sigma_j \in \Sigma^*$. Then L is context-free if, and only if, the set $\{(n_1, \ldots, n_k) \in \mathbb{N}^k \mid \sigma_1^{n_1} \cdots \sigma_k^{n_k} \in L\}$ is a finite union of stratifiable linear sets.*

Example III.3. Consider the alphabet $\Sigma = \{a, b, c, d\}$. The language $\{a^n b^m c^n \mid n, m \in \mathbb{N}\}$ is context-free since the set $\{(n_1, n_2, n_3) \in \mathbb{N}^3 \mid n_1 = n_3\}$ is the periodic set generated by $\{(1,0,1), (0,1,0)\}$, and the latter is stratified. The language $\{a^n b^m c^n d^m \mid n, m \in \mathbb{N}\}$ is not context-free. This means that the semilinear set $\mathbb{N}(1,0,1,0) + \mathbb{N}(0,1,0,1)$ cannot be decomposed into a finite union of stratifiable linear sets. □

Remark III.4. Deciding whether a given semilinear set can be decomposed into a finite union of stratifiable linear sets is, to our knowledge, still open [11, p. 163].

IV. Matching Pairs And Iterable Pairs

Push and pop actions of pushdown automata can be matched in a natural way. We lift this idea to actions of VAS and, more generally, to vectors in $\mathbb{Q}^d$. A pair $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ of vectors in $\mathbb{Q}^d$ such that $\boldsymbol{v}_1 \geq \boldsymbol{0}$ and $\boldsymbol{v}_2 \not\geq \boldsymbol{0}$ is called a *matching pair*. An *iterable pair* is a matching pair $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ such that $\|\boldsymbol{v}_2\|^- \subseteq \|\boldsymbol{v}_1\|^+$. For every matching pair $(\boldsymbol{v}_1, \boldsymbol{v}_2)$, there exists a maximal non-negative rational number $\lambda \geq 0$ such that $\boldsymbol{v}_1 + \lambda \boldsymbol{v}_2 \geq \boldsymbol{0}$. We call this rational number the *ratio* of the matching pair $(\boldsymbol{v}_1, \boldsymbol{v}_2)$, and we denote it by $\mathsf{rat}(\boldsymbol{v}_1, \boldsymbol{v}_2)$. Observe that a matching pair $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ is an iterable pair if, and only if, $\mathsf{rat}(\boldsymbol{v}_1, \boldsymbol{v}_2) > 0$.

We introduce the following vector called the *matching sum* of $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ where $\lambda = \mathsf{rat}(\boldsymbol{v}_1, \boldsymbol{v}_2)$:

$$\mathsf{mat}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \begin{cases} (1-\lambda) \cdot \boldsymbol{v}_2 & \text{if } \lambda < 1 \\ (1 - \frac{1}{\lambda}) \cdot \boldsymbol{v}_1 & \text{if } \lambda \geq 1 \end{cases}$$

The following equalities show that $\boldsymbol{v}_1 + \boldsymbol{v}_2 \geq \mathsf{mat}(\boldsymbol{v}_1, \boldsymbol{v}_2)$:

$$\boldsymbol{v}_1 + \boldsymbol{v}_2 - \mathsf{mat}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \begin{cases} \boldsymbol{v}_1 + \lambda \cdot \boldsymbol{v}_2 & \text{if } \lambda < 1 \\ \frac{1}{\lambda} \cdot (\boldsymbol{v}_1 + \lambda \cdot \boldsymbol{v}_2) & \text{if } \lambda \geq 1 \end{cases}$$

Intuitively the matching sum of $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ is an under-approximation of the sum $\boldsymbol{v}_1 + \boldsymbol{v}_2$ that is in the direction of $\boldsymbol{v}_1$ or $\boldsymbol{v}_2$. The precision of this approximation is quantified by introducing the vector $\mathsf{rem}(\boldsymbol{v}_1, \boldsymbol{v}_2)$, called the *remainder* of $(\boldsymbol{v}_1, \boldsymbol{v}_2)$, defined by:

$$\mathsf{rem}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \boldsymbol{v}_1 + \boldsymbol{v}_2 - \mathsf{mat}(\boldsymbol{v}_1, \boldsymbol{v}_2)$$

Note that $\mathsf{rem}(\boldsymbol{v}_1, \boldsymbol{v}_2) \geq \boldsymbol{0}$.

Example IV.1. Let $\boldsymbol{v}_1 = (1,2,1)$ and $\boldsymbol{v}_2 = (-1,-3,2)$. We have $\mathsf{rat}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \frac{2}{3}$, $\mathsf{mat}(\boldsymbol{v}_1, \boldsymbol{v}_2) = (-\frac{1}{3}, -1, \frac{2}{3})$, and $\mathsf{rem}(\boldsymbol{v}_1, \boldsymbol{v}_2) = (\frac{1}{3}, 0, \frac{7}{3})$. □

[2]A linear subset of $\mathbb{N}^k$ admits a unique decomposition of the form $\boldsymbol{b}+\boldsymbol{P}$.

V. Witnesses of Non-Context-Freeness

Given a sequence $(\sigma_1, \ldots, \sigma_k)$ of words in $\boldsymbol{A}^*$ (implicit in the sequel), a pair (s,t) of indexes $1 \leq s < t \leq k$ is said to be *matching* (resp. *iterable*) if the pair $(\Delta(\sigma_s), \Delta(\sigma_t))$ is *matching* (resp. *iterable*). A *matching scheme* (resp. an *iterable scheme*) is a nested relation R over $\{1, \ldots, k\}$ such that every pair $(s,t) \in R$ is matching (resp. *iterable*). The following vector is called the *excess vector* of the matching scheme R:

$$\mathsf{exc}(R) = \sum_{(s,t) \in R} \Delta(\sigma_s) + \mathsf{rat}(s,t)\Delta(\sigma_t)$$

where $\mathsf{rat}(s,t)$ simply denotes $\mathsf{rat}(\Delta(\sigma_s), \Delta(\sigma_t))$.

Definition V.1. *A* witness of non-context-freeness *for a VAS $\langle \boldsymbol{A}, \boldsymbol{c}_{init} \rangle$ is a tuple $(\sigma_1, \ldots, \sigma_k, U)$, where each $\sigma_j \in \boldsymbol{A}^*$ and U is a matching scheme, such that:*

$$\sigma_1 \cdots \sigma_k \text{ is a trace of } \langle \boldsymbol{A}, \boldsymbol{c}_{init} \rangle \tag{3}$$

$$\emptyset \neq \|\Delta(\sigma_k)\|^- \subseteq \|\mathsf{exc}(U)\|^+ \tag{4}$$

$$\text{For all } (s,t) \in U \text{ with } t < k \text{ there exists an iterable pair } (s',t) \in U \text{ with } s' \leq s \tag{5}$$

In this section, we show the following proposition (other results proved in this section are not used in the sequel).

Proposition V.2. *If there exists a witness of non-context-freeness $(\sigma_1, \ldots, \sigma_k, U)$ for a VAS $\langle \boldsymbol{A}, \boldsymbol{c}_{init} \rangle$ then the trace language $\mathcal{T}(\boldsymbol{c}_{init}) \cap \sigma_1^* \cdots \sigma_k^*$ is not context-free.*

We prove the proposition by contradiction. Consider a sequence $\sigma_1, \ldots, \sigma_k$ of words in $\boldsymbol{A}^*$ such that $\sigma_1 \cdots \sigma_k$ is a trace from $\boldsymbol{c}_{\text{init}}$, and assume that $\mathcal{T}(\boldsymbol{c}_{\text{init}}) \cap \sigma_1^* \cdots \sigma_k^*$ is context-free. We introduce the set $\boldsymbol{X}$ of vectors $(x_1, \ldots, x_k) \in \mathbb{Q}_{\geq 0}^k$ such that $x_1\Delta(\sigma_1) + \cdots + x_j\Delta(\sigma_j) \geq \boldsymbol{0}$ for every $1 \leq j \leq k$. Vectors in $\boldsymbol{X}$ can be decomposed in a nested way as follows. Given a matching pair $r = (s,t)$, we introduce the vector $\boldsymbol{m}_r$ defined by:

$$\boldsymbol{m}_r = \boldsymbol{e}_s + \mathsf{rat}(s,t)\boldsymbol{e}_t$$

Observe that $\boldsymbol{m}_r \in \boldsymbol{X}$. The following lemma provides a decomposition of vectors in $\boldsymbol{X}$ thanks to iterable schemes.

Lemma V.3. *For every $\boldsymbol{x} \in \boldsymbol{X}$ there exists an iterable scheme R such that:*

$$\boldsymbol{x} \in \sum_{j \mid \Delta(\sigma_j) \geq \boldsymbol{0}} \mathbb{Q}_{\geq 0}\boldsymbol{e}_j + \sum_{r \in R} \mathbb{Q}_{\geq 0}\boldsymbol{m}_r$$

Proof: According to Theorem III.2, the following set is a finite union of stratifiable linear sets:

$$\boldsymbol{N} = \{(n_1, \ldots, n_k) \in \mathbb{N}^k \mid \sigma_1^{n_1} \cdots \sigma_k^{n_k} \in \mathcal{T}(\boldsymbol{c}_{\text{init}})\}$$

Let $\boldsymbol{x} = (x_1, \ldots, x_k)$ in $\boldsymbol{X}$. To prove the lemma, we may assume, w.l.o.g., that $\boldsymbol{x} \in \mathbb{N}^k$. Since $\sigma_1 \cdots \sigma_k$ is a trace from $\boldsymbol{c}_{\text{init}}$, we deduce, by monotonicity, that for every $n \in \mathbb{N}$:

$$\sigma_1^{1+nx_1} \cdots \sigma_k^{1+nx_k}$$

is also a trace from $\boldsymbol{c}_{\text{init}}$. Hence $(1, \ldots, 1) + n\boldsymbol{x} \in \boldsymbol{N}$ for every $n \in \mathbb{N}$. Thanks to the pigeonhole principle, there exists

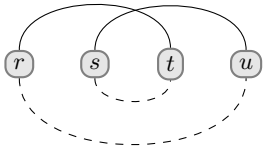

Figure 1. Lemma V.4 assumes that there exist edges (r,t) and (s,u), depicted on the top half, that are iterable pairs. In this case edges (r,u) and (s,t) are iterable pairs, depicted on the bottom half.

a stratifiable linear set that contains $(1,\dots,1)+n\boldsymbol{x}$ for an infinite number of possible $n \in \mathbb{N}$. Hence, there exist a stratifiable periodic set $\boldsymbol{P}$ and a vector $\boldsymbol{b}=(b_1,\dots,b_k)\in\mathbb{N}^k$, such that $(1,\dots,1)+n\boldsymbol{x} \in (\boldsymbol{b}+\boldsymbol{P})$ for infinitely many $n\in\mathbb{N}$. Dickson's lemma entails that $\boldsymbol{x}\in\mathbb{Q}_{\geq 0}\boldsymbol{P}$.

Since $\boldsymbol{P}$ is stratifiable, it is generated by a finite stratified set $\boldsymbol{G}$. Let us prove that $\boldsymbol{G}\subseteq\boldsymbol{X}$. Pick a vector $\boldsymbol{g}=(g_1,\dots,g_k)$ in $\boldsymbol{G}$. As $\boldsymbol{b}+\mathbb{N}\boldsymbol{g}\subseteq\boldsymbol{N}$ we deduce that, for every $n\in\mathbb{N}$, the word

$$\sigma_1^{b_1+ng_1}\cdots\sigma_k^{b_k+ng_k}$$

is a trace from $\boldsymbol{c}_{\text{init}}$. Hence, $\boldsymbol{c}_{\text{init}}+\Delta(\sigma_1^{b_1+ng_1}\cdots\sigma_j^{b_j+ng_j})\geq\boldsymbol{0}$ for every $1\leq j\leq k$ and $n\in\mathbb{N}$. Note that this vector is equal to $\boldsymbol{c}_{\text{init}}+\Delta(\sigma_1^{b_1}\cdots\sigma_j^{b_j})+n\boldsymbol{v}_j$ where $\boldsymbol{v}_j=g_1\Delta(\sigma_1)+\cdots+g_j\Delta(\sigma_j)$. We deduce that $\boldsymbol{v}_j\geq-\frac{1}{n}\cdot(\boldsymbol{c}_{\text{init}}+\Delta(\sigma_1^{b_1}\cdots\sigma_j^{b_j}))$. Thus $\boldsymbol{v}_j\geq\boldsymbol{0}$. We have proved that $\boldsymbol{g}\in\boldsymbol{X}$.

As $\boldsymbol{G}$ is stratified, there exists a nested relation U over $\{1,\dots,k\}$ such that $\boldsymbol{G}\subseteq\bigcup_{(s,t)\in U}\mathbb{Q}_{\geq 0}\boldsymbol{e}_s+\mathbb{Q}_{\geq 0}\boldsymbol{e}_t$. Let R be the set of pairs $(s,t)\in U$ that are iterable. We consider the following set:

$$\boldsymbol{C}=\sum_{j\mid\Delta(\sigma_j)\geq\boldsymbol{0}}\mathbb{Q}_{\geq 0}\boldsymbol{e}_j+\sum_{r\in R}\mathbb{Q}_{\geq 0}\boldsymbol{m}_r$$

Now let us prove that $\boldsymbol{G}\subseteq\boldsymbol{C}$. Let $\boldsymbol{g}\in\boldsymbol{G}$. There exists $(s,t)\in U$ such that $\boldsymbol{g}\in\mathbb{Q}_{\geq 0}\boldsymbol{e}_s+\mathbb{Q}_{\geq 0}\boldsymbol{e}_t$. Assume first that $s=t$. Since $\boldsymbol{g}\in\boldsymbol{X}$, we deduce that $g_s\Delta(\sigma_s)\geq\boldsymbol{0}$. If $\Delta(\sigma_s)\geq\boldsymbol{0}$ then $\boldsymbol{g}\in\boldsymbol{C}$, and if $\Delta(\sigma_s)\not\geq\boldsymbol{0}$ then $g_s=0$, which entails that $\boldsymbol{g}=\boldsymbol{0}\in\boldsymbol{C}$. Thus, in both cases we get $\boldsymbol{g}\in\boldsymbol{C}$. Next assume that $s<t$. Since $\boldsymbol{g}\in\boldsymbol{X}$ we deduce that $g_s\Delta(\sigma_s)\geq\boldsymbol{0}$ and $g_s\Delta(\sigma_s)+g_t\Delta(\sigma_t)\geq\boldsymbol{0}$. If $g_s=0$ or $g_t=0$, we deduce as previously that $\boldsymbol{g}\in\boldsymbol{C}$. So, we can assume that $g_s>0$ and $g_t>0$. From $g_s\Delta(\sigma_s)\geq\boldsymbol{0}$, we get $\Delta(\sigma_s)\geq\boldsymbol{0}$. Note that if $\Delta(\sigma_t)\geq\boldsymbol{0}$ then $\boldsymbol{g}\in\boldsymbol{C}$. So, we can assume that $\Delta(\sigma_t)\not\geq\boldsymbol{0}$. In this case $(\Delta(\sigma_s),\Delta(\sigma_t))$ is a matching pair. From $g_s\Delta(\sigma_s)+g_t\Delta(\sigma_t)\geq\boldsymbol{0}$ we get $\Delta(\sigma_s)+\frac{g_t}{g_s}\Delta(\sigma_t)\geq\boldsymbol{0}$. By maximality of the ratio, we get $\mathsf{rat}(s,t)\geq\frac{g_t}{g_s}$. In particular, $\mathsf{rat}(s,t)>0$. We deduce that (s,t) is an iterable pair. Thus $(s,t)\in R$. The equality $\boldsymbol{g}=(g_s-\frac{g_t}{\mathsf{rat}(s,t)})\boldsymbol{e}_s+\frac{g_t}{\mathsf{rat}(s,t)}(\boldsymbol{e}_s+\mathsf{rat}(s,t)\boldsymbol{e}_t)$ shows that $\boldsymbol{g}\in\boldsymbol{C}$.

Recall that $\boldsymbol{x}\in\mathbb{Q}_{\geq 0}\boldsymbol{P}$ and $\boldsymbol{P}$ is generated by $\boldsymbol{G}$. We deduce that $\boldsymbol{x}$ is a finite sum of vectors in $\mathbb{Q}_{\geq 0}\boldsymbol{G}$. Thus $\boldsymbol{x}\in\boldsymbol{C}$, since $\boldsymbol{G}\subseteq\boldsymbol{C}$, $\mathbb{Q}_{\geq 0}\boldsymbol{C}\subseteq\boldsymbol{C}$, and $\boldsymbol{C}$ is stable by finite sums. ∎

We deduce the following lemma, which is depicted in Figure 1.

Lemma V.4. *For every $1\leq r\leq s<t\leq u\leq k$ such that (r,t) and (s,u) are iterable pairs then (r,u) and (s,t) are iterable pairs.*

Proof: If $r=s$ or $t=u$ the lemma is immediate. So, we can assume, w.l.o.g., that $r<s$ and $t<u$. We introduce the vector $\boldsymbol{x}$ defined by:

$$\boldsymbol{x}\ =\boldsymbol{m}_{r,t}\ +\ \boldsymbol{m}_{s,u} \tag{6}$$

Observe that $\boldsymbol{x}\in\boldsymbol{X}$. We derive from Lemma V.3 that there exists an iterable scheme $R\subseteq\{r,s\}\times\{t,u\}$ such that $\boldsymbol{x}\in\mathbb{Q}_{\geq 0}\boldsymbol{e}_r+\mathbb{Q}_{\geq 0}\boldsymbol{e}_s+\sum_{(i,j)\in R}\mathbb{Q}_{\geq 0}\boldsymbol{m}_{i,j}$. We obtain that $\boldsymbol{x}$ may be written as

$$\boldsymbol{x}\ =\ \sum_{i\in\{r,s\}}\alpha_i\cdot\boldsymbol{e}_i\ +\ \sum_{(i,j)\in R}\alpha_{i,j}\cdot(\boldsymbol{e}_i+\mathsf{rat}(i,j)\cdot\boldsymbol{e}_j) \tag{7}$$

where the α_i and $\alpha_{i,j}$ are nonnegative rational numbers such that $\alpha_{i,j}=0$ when $(i,j)\notin R$. We derive from (6) and (7) that the α_i and $\alpha_{i,j}$ satisfy the following system of equations:

$$\begin{cases}\alpha_r+\alpha_{r,u} &= 1-\alpha_{r,t}\\ \alpha_s+\alpha_{s,t} &= 1-\alpha_{s,u}\\ \alpha_{s,t}\cdot\mathsf{rat}(s,t) &= (1-\alpha_{r,t})\cdot\mathsf{rat}(r,t)\\ \alpha_{r,u}\cdot\mathsf{rat}(r,u) &= (1-\alpha_{s,u})\cdot\mathsf{rat}(s,u)\end{cases}$$

Recall that $\mathsf{rat}(r,t)>0$ and $\mathsf{rat}(s,u)>0$ since (r,t) and (s,u) are iterable pairs. Moreover, since R is nested, $\alpha_{r,t}=0$ or $\alpha_{s,u}=0$. We derive from the above system of equations that $\mathsf{rat}(r,u)>0$ and $\mathsf{rat}(s,t)>0$. Consequently, the pairs (r,u) and (s,t) are iterable pairs. ∎

Now, let us assume, by contradiction, that there exists a matching scheme U such that $(\sigma_1,\dots,\sigma_k,U)$ is a witness of non-context-freeness for $\langle\boldsymbol{A},\boldsymbol{c}_{\text{init}}\rangle$. In this case there exists $\mu>0$ such that $\mathsf{exc}(U)+\mu\Delta(\sigma_k)\geq\boldsymbol{0}$. We introduce the vector $\boldsymbol{x}$ defined by:

$$\boldsymbol{x}=\mu\boldsymbol{e}_k+\sum_{u\in U}\boldsymbol{m}_u \tag{8}$$

Observe that $\boldsymbol{x}\in\boldsymbol{X}$. We derive from Lemma V.3 that there exists an iterable scheme R such that:

$$\boldsymbol{x}\in\sum_{j\mid\Delta(\sigma_j)\geq\boldsymbol{0}}\mathbb{Q}_{\geq 0}\boldsymbol{e}_j+\sum_{r\in R}\mathbb{Q}_{\geq 0}\boldsymbol{m}_r \tag{9}$$

From (8) and (9) we deduce that there exists $\boldsymbol{z}\in\sum_{j\mid\Delta(\sigma_j)\geq\boldsymbol{0}}\mathbb{Q}_{\geq 0}\boldsymbol{e}_j$ such that:

$$\mu\boldsymbol{e}_k+\sum_{u\in U}\boldsymbol{m}_u\ \in\ \boldsymbol{z}+\sum_{r\in R}\mathbb{Q}_{\geq 0}\boldsymbol{m}_r$$

By removing from the previous membership vectors $\boldsymbol{m}_{s,t}$ occurring in both sides, we deduce two sequences $(\alpha_u)_{u\in U}$ and $(\beta_r)_{r\in R}$ of non-negative rational numbers such that $\alpha_{s,t}\beta_{s,t}=0$ for every $(s,t)\in U\cap R$ and such that:

$$\mu\boldsymbol{e}_k+\sum_{u\in U}\alpha_u\cdot\boldsymbol{m}_u\ =\ \boldsymbol{z}+\sum_{r\in R}\beta_r\cdot\boldsymbol{m}_r \tag{10}$$

We consider the relation $\tilde{U}$ of iterable pairs $u\in U$ such that $\alpha_u>0$ and the set $\tilde{R}$ of pairs $r\in R$ such that $\beta_r>0$ (these

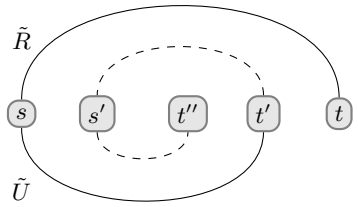

Figure 2. Iteration pairs (s', t') and (s', t'') constructed in the proof of Lemma V.6. Edges are pairs $(s, t), (s', t') \in \tilde{R}$ on the top half, and $(s, t'), (s', t'') \in \tilde{U}$ on the bottom half.

pairs are iterable since R is an iterable scheme). We will need the two following lemmas, which are depicted in Figure 2.

Lemma V.5. *There exists* $1 \leq s < t' < t \leq k$ *such that* $(s, t) \in \tilde{R}$ *and* $(s, t') \in \tilde{U}$.

Proof: We consider $t = k$. Since $\mu > 0$, we deduce from (10) that there exists $r \in R$ such that $\beta_r > 0$ and $\boldsymbol{m}_r(k) > 0$. Hence, there exists $s < k$ such that $r = (s, k)$. Note that $(s, k) \in \tilde{R}$. Since $\beta_r > 0$ and $\boldsymbol{m}_r(s) > 0$, we derive from (10) that there exists $u \in U$ such that $\alpha_u > 0$ and $\boldsymbol{m}_u(s) > 0$. There exists $t' > s$ such that $u = (s, t')$. Note that $t' = k$ implies $u = r$ and $\alpha_u \beta_r > 0$ which is impossible. Thus $t' < k$. Since U is a witness of non-context-freeness, we derive from (5) that there exists $s_0 \leq s$ such that (s_0, t') is an iterable pair. Since $s_0 \leq s < t' < t$ and (s_0, t') and (s, t) are iterable pairs, Lemma V.4 shows that (s, t') is an iterable pair. Thus $(s, t') \in \tilde{U}$. ∎

Lemma V.6. *For every* $1 \leq s < t' < t \leq k$ *such that* $(s, t) \in \tilde{R}$ *and* $(s, t') \in \tilde{U}$*, there exist* $s < s' < t'' < t'$ *such that* $(s', t') \in \tilde{R}$ *and* $(s', t'') \in \tilde{U}$.

Proof: From $(s, t') \in \tilde{U}$ we get $\alpha_{s,t'} > 0$ and $\boldsymbol{m}_{s,t'}(t') > 0$. As $\Delta(\sigma_{t'}) \not\geq \mathbf{0}$ we get $\boldsymbol{z}(t') = 0$. From (10) we deduce that there exists $r \in R$ such that $\beta_r > 0$ and $\boldsymbol{m}_r(t') > 0$. Hence $r \in \tilde{R}$ and there exists $s' < t'$ such that $r = (s', t')$. Since R is nested and $(s, t) \in R$ we deduce that $s \leq s'$. Note that $s = s'$ implies $\beta_r = \beta_{s,t'} > 0$ which is in contradiction with $\alpha_{s,t'} \beta_{s,t'} = 0$. Thus $s < s' < t' < t$. Since $\Delta(\sigma_{s'}) \geq \mathbf{0}$ we get $\mu \boldsymbol{e}_k(s') = 0$. Since $\beta_r > 0$ and $\boldsymbol{m}_r(s') > 0$, we deduce from (10) that there exists $u \in U$ such that $\alpha_u > 0$ and $\boldsymbol{m}_u(s') > 0$. Hence, there exists $t'' > s'$ such that $u = (s', t'')$. Note that if $t' < t''$ then $s < s' < t' < t''$ which is impossible since $(s, t'), (s', t'') \in U$ and U is nested. Thus $t'' \leq t'$. Note that $t'' = t'$ implies $\alpha_u = \alpha_{s',t'} > 0$ which is in contradiction with $\alpha_{s',t'} \beta_{s',t'} = 0$. Thus $t'' < t'$. In particular $t'' < k$ and since U is a witness, we derive from (5) that there exists $s_0 \leq s'$ such that (s_0, t'') is an iterable pair. Since (s_0, t'') and (s', t') are iterable pairs and $s_0 \leq s' < t'' < t'$, Lemma V.4 shows that (s', t'') is an iterable pair. Thus $(s', t'') \in \tilde{U}$. ∎

From the two previous lemmas, we deduce an infinite sequence $(s_i, t_i)_{i \geq 1}$ such that $1 \leq s_i < t_{i+1} < t_i \leq k$, $(s_i, t_i) \in \tilde{R}$, and $(s_i, t_{i+1}) \in \tilde{U}$. Informally, this means that the picture of Figure 2 can be iterated to produce an infinite spiral. We get a contradiction since the set $\{1, \ldots, k\}$ is finite. This concludes the proof of Proposition V.2.

VI. Vector Pushdown Automata

We introduce an extension of pushdown automata that will be convenient in the next section to simulate the behavior of VAS. Informally, vector pushdown automata are finite-state automata equipped with two unbounded storage devices: a counter $\mathtt{r}$ holding a vector in $\mathbb{Q}^d_{\geq 0}$, and a pushdown stack $\mathtt{z}$, where each stack symbol is a vector in $\mathbb{Q}^d_{\geq 0}$. Actions on the counter are limited to translations by a vector in $\mathbb{Q}^d_{\geq 0}$, and actions on the stack are the usual push and pop operations. In addition, the automaton may test linear constraints, with nonnegative coefficients, involving its counter and the sum of all stacked vectors. Formally, we define the set $\mathtt{Op} = \mathtt{Op}_{cnt} \cup \mathtt{Op}_{lifo} \cup \mathtt{Op}_{test}$ of operations by

$$\begin{aligned} \mathtt{Op}_{cnt} &= \{\mathsf{add}(\boldsymbol{v}) \mid \boldsymbol{v} \in \mathbb{Q}^d_{\geq 0}\} \\ \mathtt{Op}_{lifo} &= \{\mathsf{push}(\boldsymbol{\gamma}), \mathsf{pop}(\boldsymbol{\gamma}) \mid \boldsymbol{\gamma} \in \mathbb{Q}^d_{\geq 0}\} \\ \mathtt{Op}_{test} &= \{\mathsf{test}(\varphi) \mid \varphi \in \Phi\} \end{aligned}$$

where Φ is the set of all Boolean combinations of constraints

$$\sum_{i=1}^{d} \alpha_i \mathtt{r}(i) + \sum_{i=1}^{d} \beta_i \Delta(\mathtt{z})(i) \ \# \ c$$

with $\alpha_i, \beta_i \in \mathbb{N}$, $\# \in \{\leq, \geq\}$, and $c \in \mathbb{Z}$. It is understood that a formula $\varphi \in \Phi$ has free variables $\mathtt{r}$ and $\mathtt{z}$.

A *vector pushdown automaton* is a 5-tuple $\mathcal{P} = \langle Q, q_{\text{init}}, \Sigma, T \rangle$ where Q is a set of *states*, $q_{\text{init}} \in Q$ is the *initial* state, Σ is the *input alphabet*, and $T \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathtt{Op} \times Q$ is a set of *transitions*. There is no explicit set of final states as we won't need them. Note, also, that the sets Q and T may be infinite. A transition $(q, \ell, \mathtt{op}, q')$, written $q \xrightarrow[\ell]{\mathtt{op}} q'$, means that the automaton moves from state q to state q', by reading the input letter $\ell \in (\Sigma \cup \{\varepsilon\})$ and performing $\mathtt{op}$. We call ℓ the *label* of t, and $\mathtt{op}$ its *operation*.

We give the operational semantics of $\mathcal{P}$ as a labeled transition system. The set of *configurations* is $\mathcal{C} = Q \times \mathbb{Q}^d_{\geq 0} \times (\mathbb{Q}^d_{\geq 0})^*$, and the *initial configuration* is $(q_{\text{init}}, \mathbf{0}, \varepsilon)$. The *step relation* of a transition $t = (q, \ell, \mathtt{op}, q')$, written $\xrightarrow{t}$, is the least binary relation on $\mathcal{C}$ satisfying the following conditions:

$$\begin{aligned} (q, \boldsymbol{r}, z) &\xrightarrow{t} (q', \boldsymbol{r} + \boldsymbol{v}, z) && \text{if} \quad \mathtt{op} = \mathsf{add}(\boldsymbol{v}) \\ (q, \boldsymbol{r}, z) &\xrightarrow{t} (q', \boldsymbol{r}, z\boldsymbol{\gamma}) && \text{if} \quad \mathtt{op} = \mathsf{push}(\boldsymbol{\gamma}) \\ (q, \boldsymbol{r}, z\boldsymbol{\gamma}) &\xrightarrow{t} (q', \boldsymbol{r}, z) && \text{if} \quad \mathtt{op} = \mathsf{pop}(\boldsymbol{\gamma}) \\ (q, \boldsymbol{r}, z) &\xrightarrow{t} (q', \boldsymbol{r}, z) && \text{if} \quad \mathtt{op} = \mathsf{test}(\varphi) \text{ and } \boldsymbol{r}, z \models \varphi \end{aligned}$$

A *run* in $\mathcal{P}$ is a finite, alternating sequence $(c_0, t_1, c_1, \ldots, t_k, c_k)$ of configurations $c_i \in \mathcal{C}$ and transitions $t_i \in T$, satisfying $c_{i-1} \xrightarrow{t_i} c_i$ for all i. The *label* of the run is the word $\ell_1 \cdots \ell_k \in \Sigma^*$, where each ℓ_i is the label of t_i. For brevity, we will sometimes replace the transition t by its label and/or operation when dealing with steps and runs. The *language* recognized by $\mathcal{P}$ is the set of words in Σ^* that

label some run from the initial configuration. A state (resp. transition, configuration) is called *reachable* in $\mathcal{P}$ when it occurs on some run from the initial configuration.

Proposition VI.1. *The language recognized by a vector pushdown automaton with finitely many states and transitions is effectively context-free.*

Proof: Let $\mathcal{P} = \langle Q, q_{\text{init}}, \Sigma, T\rangle$ be a vector pushdown automaton, and assume that Q and T are finite. Observe that the language recognized by $\mathcal{P}$ is preserved when every vector occurring in the operations of T is multiplied by the same positive natural number. So we may assume, w.l.o.g., that every vector in the operations of T is in $\mathbb{N}^d$. To prove that the language recognized by $\mathcal{P}$ is context-free, we construct from $\mathcal{P}$ a new vector pushdown automaton $\mathcal{Q}$ without $\mathsf{add}(\boldsymbol{v})$ nor $\mathsf{test}(\varphi)$ operations. Let us define K to be the maximal constant c occurring in $\mathsf{test}(\varphi)$ operations of T. The constant K acts as a threshold to abstract large components of the counter and of the sum of stacked vectors. Formally, the *abstraction* of a vector $\boldsymbol{x} \in \mathbb{N}^d$ is the vector $\boldsymbol{x}^\sharp$ in $(\{0,\dots,K\}\cup\{\star\})^d$, defined by $\boldsymbol{x}^\sharp(i) = \star$ if $\boldsymbol{x}(i) > K$ and $\boldsymbol{x}^\sharp(i) = \boldsymbol{x}(i)$ otherwise. The counter of $\mathcal{P}$ is replaced by its abstraction in $\mathcal{Q}$. This is possible because all operations are monotonic w.r.t. the counter. As $\mathcal{Q}$'s abstracted counter may take only finitely values, we store it as part of the state. We also maintain, in $\mathcal{Q}$, the abstraction of the sum of all stacked vectors. To this end, $\mathcal{Q}$ encodes a stack content $\boldsymbol{\gamma}_1\cdots\boldsymbol{\gamma}_h$ of $\mathcal{P}$ by $\boldsymbol{\gamma}_1\boldsymbol{s}_1\cdots\boldsymbol{\gamma}_h\boldsymbol{s}_h$, where each $\boldsymbol{s}_i$ satisfies $\boldsymbol{s}_i = (\sum_{j=1}^{i}\boldsymbol{\gamma}_j)^\sharp$. Remark that $\mathcal{Q}$ needs only finitely many additional states and transitions to maintain this encoding. Obviously, the information provided by the vector $\boldsymbol{s}$ on top of the stack is sufficient (together with the abstracted counter) to faithfully simulate the $\mathsf{test}(\varphi)$ operations of $\mathcal{P}$. We obtain that $\mathcal{P}$ and $\mathcal{Q}$ recognize the same language. Since $\mathcal{Q}$ contains only stack operations in Op_{lifo} and has finitely many states and transitions, the language that it recognizes is context-free. ∎

VII. Simulating VAS with Vector Pushdown Automata

For the remainder of the paper, we assume a fixed VAS $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}}\rangle$. Let us call any subset W of $\boldsymbol{A}^*$ that is non-empty and prefix-closed a *support set*. We introduce a vector pushdown automaton $\mathcal{P}_W$ that simulates the behavior of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}}\rangle$. The simulation is parameterized by a support set W. As will be clear later, the support set W has the effect of restricting the behavior of $\mathcal{P}_W$. In particular, $\mathcal{P}_W$ will be shown to have finitely many reachable states and transitions when W is finite. However, the support set W need not be finite, in general.

Formally, given a support set $W \subseteq \boldsymbol{A}^*$, the vector pushdown automaton $\mathcal{P}_W$, with input alphabet $\boldsymbol{A}$, is defined as follows. States of $\mathcal{P}_W$ are pairs $q = (\boldsymbol{p}, w)$ where $\boldsymbol{p}$ is a vector in $\mathbb{Q}^d$ and w is a word in $W \cup W\boldsymbol{A}$. In addition, two sink states $\bot$ and $\lightning$ are introduced to model failures of the simulation. The initial state is $(\boldsymbol{0}, \varepsilon)$. The transitions of $\mathcal{P}_W$ are formally defined hereafter, but, first, let us explain its behavior informally. There are two operational modes, depending on the state's first component $\boldsymbol{p}$. If $\boldsymbol{p} = \boldsymbol{0}$ then $\mathcal{P}_W$ is *idle*, i.e., ready to read an input symbol. Otherwise, $\mathcal{P}_W$ is *processing* the vector $\boldsymbol{p}$. Let us describe a run of $\mathcal{P}_W$ over an input word u that is a trace of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}}\rangle$. Initially, $\mathcal{P}_W$ is in the state $(\boldsymbol{0}, \varepsilon)$, with a zero counter and an empty stack. First, $\mathcal{P}_W$ reads input symbols and appends them to the support w that is part of the state. As soon as the support ends with a cycle[3] $\sigma \neq \varepsilon$ such that $\Delta(\sigma) \geq \boldsymbol{0}$, $\mathcal{P}_W$ extracts the cycle σ from the support, pushes $\Delta(\sigma)$ on the stack, and moves to the state $(\boldsymbol{0}, w')$ where $w = w'\sigma$. Then, $\mathcal{P}_W$ resumes its computation: it appends the input to the support and extracts cycles. However, extraction of cycles receives help from the stack: the condition $\Delta(\sigma) \geq \boldsymbol{0}$ becomes $\boldsymbol{s} + \Delta(\sigma) \geq \boldsymbol{0}$, where $\boldsymbol{s}$ is the sum of all previously extracted cycles. When $\mathcal{P}_W$ extracts a cycle σ with $\Delta(\sigma) \not\geq \boldsymbol{0}$, it moves to the processing state $(\Delta(\sigma), w')$ instead of pushing $\Delta(\sigma)$ on the stack. Then, vectors are popped from the stack and used to match $\Delta(\sigma)$. Each time, the remainder is added to the counter of $\mathcal{P}_W$. If $\Delta(\sigma)$ gets fully matched with the stack, $\mathcal{P}_W$ moves to the idle state $(\boldsymbol{0}, w')$. If, on the contrary, the stack becomes empty before $\Delta(\sigma)$ is fully processed, $\mathcal{P}_W$ moves to the failure state $\lightning$. Another cause for failure is when the component w of the state gets outside of the support set W (after reading an input symbol). In that case, $\mathcal{P}_W$ moves to the failure state $\bot$.

We now make the above ideas more precise. To simplify the presentation, we use a couple of notational shortcuts for formulas φ used in $\mathsf{test}(\varphi)$ operations. Given a vector $\boldsymbol{v} \in \mathbb{Z}^d$ and $\# \in \{\leq, \geq\}$,

$$\mathtt{r} + \Delta(\mathtt{z}) + \boldsymbol{v} \,\#\, \boldsymbol{0} \quad \text{stands for} \quad \bigwedge_{i=1}^{d} \mathtt{r}(i) + \Delta(\mathtt{z})(i) \,\#\, -\boldsymbol{v}(i)$$

$$\|\boldsymbol{v}\|^+ \subseteq \|\mathtt{r}\|^+ \quad \text{stands for} \quad \bigwedge_{\boldsymbol{v}(i)>0} \neg(\mathtt{r}(i) \leq 0)$$

The formula $\mathit{Extr}(w)$ used in the first rule specifies that an "extract cycle" transition (see the third rule) can be taken. This formula is defined by

$$\mathit{Extr}(w) \;=\; \bigvee_{\sigma\neq\varepsilon \text{ suffix of } w} \mathtt{r} + \Delta(\mathtt{z}) + \Delta(\sigma) \;\geq\; \boldsymbol{0}$$

Formally, the (infinite) set of transitions of $\mathcal{P}_W$ is given by the following rules, where $\boldsymbol{p}$ ranges over $\mathbb{Q}^d$, w ranges over W, and $\boldsymbol{\gamma}$ ranges over $\mathbb{Q}^d_{\geq 0}$.

- *Read an input vector.* For every vector $\boldsymbol{a} \in \boldsymbol{A}$,

$$(\boldsymbol{0}, w) \xrightarrow{\mathsf{test}(\neg \mathit{Extr}(w))} \cdot \xrightarrow[\boldsymbol{a}]{\mathsf{test}(\mathtt{r}+\Delta(\mathtt{z})+\boldsymbol{v}\geq\boldsymbol{0})} (\boldsymbol{0}, w\boldsymbol{a})$$

  where $\boldsymbol{v} = \boldsymbol{c}_{\text{init}} + \Delta(w) + \boldsymbol{a}$.
- *Fail with missing support.* If $w \in W\boldsymbol{A} \setminus W$,

$$(\boldsymbol{0}, w) \to \bot$$

[3] In our setting, a cycle is nothing more than a non-empty sequence of actions in $\boldsymbol{A}^*$. We use the term cycle since the suffixes that are extracted by $\mathcal{P}_W$ would be cycles for a vector addition system with states.

- *Extract cycle*. For every word $\sigma \in \boldsymbol{A}^*$ with $w\sigma \in W$ and $\sigma \neq \varepsilon$,

$$(\mathbf{0}, w\sigma) \xrightarrow{\mathsf{test}(\mathtt{r}+\Delta(\mathtt{z})+\Delta(\sigma)\geq \mathbf{0})} (\Delta(\sigma), w)$$

- *Positive processing*. If $\boldsymbol{p} > \mathbf{0}$,

$$(\boldsymbol{p}, w) \xrightarrow{\mathsf{test}(\|\boldsymbol{p}\|^+ \subseteq \|\mathtt{r}\|^+)} \cdot \xrightarrow{\mathsf{add}(\boldsymbol{p})} (\mathbf{0}, w)$$

$$(\boldsymbol{p}, w) \xrightarrow{\mathsf{test}(\|\boldsymbol{p}\|^+ \not\subseteq \|\mathtt{r}\|^+)} \cdot \xrightarrow{\mathsf{push}(\boldsymbol{p})} (\mathbf{0}, w)$$

- *Non-positive processing*. If $\boldsymbol{p} \not\geq \mathbf{0}$,

$$(\boldsymbol{p}, w) \xrightarrow{\mathsf{pop}(\boldsymbol{\gamma})} \cdot \xrightarrow{\mathsf{add}(\mathsf{rem}(\boldsymbol{\gamma},\boldsymbol{p}))} (\mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{p}), w)$$

- *Fail with empty stack*. If $\boldsymbol{p} \not\geq \mathbf{0}$,

$$(\boldsymbol{p}, w) \xrightarrow{\mathsf{test}(\Delta(\mathtt{z})=\mathbf{0})} \lightning$$

The vector pushdown automaton $\mathcal{P}_W$ might seem useless at first glance, since it has infinitely many states and infinitely many stack symbols. But, in fact, it is the main ingredient of our upcoming algorithm (see Figure 3 page 10) solving the context-freeness problem for VAS. Before that, we need to establish a few preparatory results.

Let us first prove that $\mathcal{P}_W$ faithfully simulates the VAS $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ when it does not fail. The following notation will be helpful to formally present the simulation of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ by $\mathcal{P}_W$. Given a configuration $(\boldsymbol{p}, w, \boldsymbol{r}, z)$ of $\mathcal{P}_W$, we denote by $\mathsf{val}(\boldsymbol{p}, w, \boldsymbol{r}, z)$ its *value*, defined by

$$\mathsf{val}(\boldsymbol{p}, w, \boldsymbol{r}, z) \;=\; \boldsymbol{c}_{\text{init}} + \boldsymbol{p} + \Delta(w) + \boldsymbol{r} + \Delta(z) \qquad (11)$$

Observe that the value remains constant under the application of "extract cycle" and "processing" transitions. It follows, by a routine induction, that

$$\mathsf{val}(\boldsymbol{p}, w, \boldsymbol{r}, z) \;=\; \boldsymbol{c}_{\text{init}} + \Delta(u) \;\geq\; \mathbf{0} \qquad (12)$$

for every run $(\mathbf{0}, \varepsilon, \mathbf{0}, \varepsilon) \xrightarrow[u]{*} (\boldsymbol{p}, w, \boldsymbol{r}, z)$ in $\mathcal{P}$. Hence, the language recognized by $\mathcal{P}_W$ is contained in the trace language of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$. The reverse inclusion does not hold in general, but we provide a sufficient condition for it. Formally, we say that $\mathcal{P}_W$ *succeeds* if neither $\bot$ nor $\lightning$ is reachable in $\mathcal{P}_W$.

Lemma VII.1. *If $\mathcal{P}_W$ succeeds then it contains, for every reachable configuration $(\mathbf{0}, w, \boldsymbol{r}, z)$, a run $(\mathbf{0}, w, \boldsymbol{r}, z) \xrightarrow[\varepsilon]{*} (\mathbf{0}, w', \boldsymbol{r}', z')$ such that $\boldsymbol{r}', \Delta(z') \models \neg Extr(w')$.*

Proof: Consider a reachable configuration $(\mathbf{0}, w, \boldsymbol{r}, z)$. We show that, if $\boldsymbol{r}, \Delta(z) \models Extr(w)$, then $\mathcal{P}_W$ may reach, from the configuration $(\mathbf{0}, w, \boldsymbol{r}, z)$, a configuration with state $(\mathbf{0}, w')$ where w' is a proper prefix of w. The lemma will follow by induction on $|w|$.

Assume that $\boldsymbol{r}, \Delta(z) \models Extr(w)$. There exists a non-empty suffix σ of w such that $\boldsymbol{r} + \Delta(z) + \Delta(\sigma) \geq \mathbf{0}$. Moreover, $w \in W$ since $\bot$ is not reachable in $\mathcal{P}_W$. Hence, $\mathcal{P}_W$ may move, via an "extract cycle" transition, to a configuration with state $(\boldsymbol{p}', w')$, where $w = w'\sigma$. If $\boldsymbol{p}' \not\geq \mathbf{0}$, then $\mathcal{P}_W$ may take "non-positive processing" transitions and reach a configuration with state $(\boldsymbol{p}'', w')$ for some $\boldsymbol{p}'' \geq \mathbf{0}$. Indeed, the stack may not become empty in the process since $\mathcal{P}_W$ succeeds. So we may assume, w.l.o.g., that $\boldsymbol{p}' \geq \mathbf{0}$. If $\boldsymbol{p}' = \mathbf{0}$ then we are done. Otherwise, $\mathcal{P}_W$ may take a "positive processing" transition and reach a configuration with state $(\mathbf{0}, w')$. ∎

Proposition VII.2. *If $\mathcal{P}_W$ succeeds then it recognizes the trace language of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$.*

Proof: It remains to show that the language recognized by $\mathcal{P}_W$ contains the trace language of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$. We show by induction on $|u|$ that, for every trace u of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$, there exists a run $(\mathbf{0}, \varepsilon, \mathbf{0}, \varepsilon) \xrightarrow[u]{*} (\mathbf{0}, w, \boldsymbol{r}, z)$ in $\mathcal{P}$. The basis $u = \varepsilon$ is trivial. To prove the induction step, consider a trace $u\boldsymbol{a}$ of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$, and assume that $(\mathbf{0}, \varepsilon, \mathbf{0}, \varepsilon) \xrightarrow[u]{*} (\mathbf{0}, w, \boldsymbol{r}, z)$ is a run in $\mathcal{P}$. Since $\mathcal{P}_W$ succeeds, we derive from Lemma VII.1 that $\mathcal{P}_W$ contains a run $(\mathbf{0}, w, \boldsymbol{r}, z) \xrightarrow[\varepsilon]{*} (\mathbf{0}, w', \boldsymbol{r}', z')$ such that $\boldsymbol{r}', \Delta(z') \models \neg Extr(w')$. Observe that

$$\begin{aligned} \boldsymbol{c}_{\text{init}} + \Delta(w') + \boldsymbol{r}' + \Delta(z') + \boldsymbol{a} &= \mathsf{val}(\mathbf{0}, w', \boldsymbol{r}', z') + \boldsymbol{a} \\ &= \boldsymbol{c}_{\text{init}} + \Delta(u) + \boldsymbol{a} \\ &\geq \mathbf{0} \end{aligned}$$

Therefore, $\mathcal{P}_W$ contains a "read" transition $(\mathbf{0}, w', \boldsymbol{r}', z') \xrightarrow[a]{t} (\mathbf{0}, w'\boldsymbol{a}, \boldsymbol{r}', z')$, which concludes the proof. ∎

Now, let us prove that if $\mathcal{P}_W$ reaches the state $\lightning$ then $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ admits a witness of non-context-freeness. This witness is obtained from the sequence $(\sigma_1, \ldots, \sigma_k)$ of cycles extracted along the run and the nested relation corresponding to the matching push/pop operations.

A sequence $(\sigma_1, \ldots, \sigma_k)$ of words in $\boldsymbol{A}^*$ is said to be *compensable* if $\|\Delta(\sigma_j)\|^- \subseteq \|\Delta(\sigma_1)\|^+ \cup \cdots \cup \|\Delta(\sigma_{j-1})\|^+$ for every $1 \leq j \leq k$. In this case, the set of indexes $I = \|\Delta(\sigma_1)\|^+ \cup \cdots \cup \|\Delta(\sigma_k)\|^+$ is called the set of *increased* components.

Lemma VII.3. *For every compensable sequence $(\sigma_1, \ldots, \sigma_k)$ there exists a sequence $n_1, \ldots, n_k \in \mathbb{N}$ such that the vector $\boldsymbol{v}_j = \Delta(\sigma_1^{n_1} \cdots \sigma_j^{n_j})$ is in $\mathbb{N}^d$ for every $1 \leq j \leq k$ and such that $\|\boldsymbol{v}_k\|^+$ is the set of increased components.*

Proof: We prove the lemma by induction over k. The case $k = 0$ is immediate. Assume the property proved for a $k \in \mathbb{N}$ and let us consider a compensable sequence $(\sigma_1, \ldots, \sigma_{k+1})$. By induction, there exists $n_1, \ldots, n_k \in \mathbb{N}$ such that the vector $\boldsymbol{v}_j = \Delta(\sigma_1^{n_1} \cdots \sigma_j^{n_j})$ is in $\mathbb{N}^d$ for every $1 \leq j \leq k$ and such that $\|\boldsymbol{v}_k\|^+$ is the set of positive components of $(\sigma_1, \ldots, \sigma_k)$. Since $\|\Delta(\sigma_{k+1})\|^- \subseteq \|\Delta(\sigma_1)\|^+ \cup \cdots \cup \|\Delta(\sigma_k)\|^+ = \|\boldsymbol{v}_k\|^+$ we deduce that there exists $m \in \mathbb{N}$ such that $m \cdot \boldsymbol{v}_k + \Delta(\sigma_{k+1}) \geq \mathbf{0}$. Let us consider the sequence $n'_1, \ldots, n'_{k+1}$ defined by $n'_j = (m+1) \cdot n_j$ if $j \leq k$ and $n'_{k+1} = 1$ and observe that this sequence proves the induction. ∎

In the sequel we show that the cycles extracted along any run of $\mathcal{P}_W$ are compensable. We introduce the set $\mathbb{Q}^d_I$ of vectors $\boldsymbol{v} \in \mathbb{Q}^d$ such that $\boldsymbol{v}(i) = 0$ for every $i \notin I$, and the set $\boldsymbol{A}^*_I$ of words $u \in \boldsymbol{A}^*$ such that $\Delta(u) \in \mathbb{Q}^d_I$.

Lemma VII.4. *Let us consider $\boldsymbol{v} \in \mathbb{N}^d$ and a subset $I \subseteq \|\boldsymbol{v}\|^+$. If there exists a trace from a configuration $\boldsymbol{c}$ labeled by a word in $\boldsymbol{A}_I^* \boldsymbol{a}_1 \cdots \boldsymbol{A}_I^* \boldsymbol{a}_n$ then there exists $r \in \mathbb{N}$ and a trace from $\boldsymbol{c} + r \cdot \boldsymbol{v}$ labeled by $\boldsymbol{a}_1 \cdots \boldsymbol{a}_n$*

Proof: Let us consider words $u_1, \ldots, u_n \in \boldsymbol{A}_I^*$ such that $u_1 \boldsymbol{a}_1 \cdots u_n \boldsymbol{a}_n$ is a trace from $\boldsymbol{c}$. We consider the configuration $\boldsymbol{c}_j = \boldsymbol{c} + \Delta(u_1 \boldsymbol{a}_1 \cdots u_j \boldsymbol{a}_j)$. Since $I \subseteq \|\boldsymbol{v}\|^+$ there exists $r \in \mathbb{N}$ such that the vector $\boldsymbol{x}_j = \boldsymbol{c} + r \cdot \boldsymbol{v} + \Delta(\boldsymbol{a}_1 \cdots \boldsymbol{a}_j)$ satisfies $\boldsymbol{x}_j(i) \geq 0$ for every $i \in I$ and for every $0 \leq j \leq n$. Since $\boldsymbol{x}_j(i) = \boldsymbol{c}_j(i)$ for every $i \notin I$ we get $\boldsymbol{x}_j \in \mathbb{N}^d$. As $\boldsymbol{x}_{j-1} \xrightarrow{\boldsymbol{a}_j} \boldsymbol{x}_j$ for every $1 \leq j \leq k$, we have proved that there exists a trace from $\boldsymbol{c} + r \cdot \boldsymbol{v}$ labeled by $\boldsymbol{a}_1 \cdots \boldsymbol{a}_n$. ∎

A sequence $(\sigma_1, \ldots, \sigma_k)$ is said to be *insertable* in a trace u from $\boldsymbol{c}_{\text{init}}$ if there exist a decomposition of u into $u = u_1 \cdots u_{k+1}$ and a sequence $m_1, \ldots, m_k \in \mathbb{N}_{>0}$ such that $u_1 \sigma_1^{m_1} \cdots u_k \sigma_k^{m_k} u_{k+1}$ is a trace from $\boldsymbol{c}_{\text{init}}$.

Lemma VII.5. *For every configuration $(\boldsymbol{p}, \boldsymbol{a}_1 \cdots \boldsymbol{a}_n, \boldsymbol{r}, z)$ reachable in $\mathcal{P}_W$ by a run labeled by u, the sequence $(\sigma_1, \ldots, \sigma_k)$ of cycles extracted along the run is insertable in u, and it is compensable with a set I of increased components such that $\boldsymbol{p}, \boldsymbol{r} \in \mathbb{Q}_I^d$, $z \in (\mathbb{Q}_I^d)^*$, and such that:*

$$u \in \boldsymbol{A}_I^* \boldsymbol{a}_1 \cdots \boldsymbol{A}_I^* \boldsymbol{a}_n \boldsymbol{A}_I^*$$

Proof: The proof is performed by induction over the length of runs in the vector pushdown automaton. For the empty run the proof is immediate. So, let us assume that we reach a configuration $(\boldsymbol{p}, w, \boldsymbol{r}, z)$ with a run labeled by u. We denote by $(\sigma_1, \ldots, \sigma_k)$ the sequence of cycles extracted along the run. Let us consider a configuration $(\boldsymbol{p}', w', \boldsymbol{r}', z')$ reachable in one step from $(\boldsymbol{p}, w, \boldsymbol{r}, z)$ in $\mathcal{P}_W$. We assume that $(\boldsymbol{p}, w, \boldsymbol{r}, z)$ satisfies the lemma. That means $(\sigma_1, \ldots, \sigma_k)$ is insertable in u, and it is compensable with a set I of increased components such that $\boldsymbol{p}, \boldsymbol{r} \in \mathbb{Q}_I^d$, $z \in (\mathbb{Q}_I^d)^*$, and $u \in \boldsymbol{A}_I^* \boldsymbol{a}_1 \cdots \boldsymbol{A}_I^* \boldsymbol{a}_n \boldsymbol{A}_I^*$ where $w = \boldsymbol{a}_1 \cdots \boldsymbol{a}_n$.

A routine case inspection clearly shows that we only have to consider the "extract cycle" transition since the induction is immediate for the other transitions. In this case w can be decomposed into $w = w'\sigma$ where σ is a non-empty word, $\boldsymbol{p}' = \Delta(\sigma)$, $\boldsymbol{r}' = \boldsymbol{r}$, $z' = z$, and $\Delta(\sigma) + \boldsymbol{r} + \Delta(z) \geq \boldsymbol{0}$.

Since $\boldsymbol{r}(i) = 0$ and $\Delta(z)(i) = 0$ for every $i \notin I$, from $\Delta(\sigma) + \boldsymbol{r} + \Delta(z) \geq \boldsymbol{0}$ we derive that $\|\Delta(\sigma)\|^- \subseteq I$. Thus $(\sigma_1, \ldots, \sigma_k, \sigma)$ is compensable. Let us consider the set I' of components increased by this sequence and observe that $\boldsymbol{p}', \boldsymbol{r}' \in \mathbb{Q}_{I'}^d$ and $z' \in (\mathbb{Q}_{I'}^d)^*$.

Note that $u \in \boldsymbol{A}_I^* \boldsymbol{a}_1 \boldsymbol{A}_I^* \cdots \boldsymbol{a}_n \boldsymbol{A}_I^*$. Since $w = w'\sigma$, there exists $p \in \{1, \ldots, n\}$ such that $w = \boldsymbol{a}_1 \cdots \boldsymbol{a}_{p-1}$ and $\sigma = \boldsymbol{a}_p \cdots \boldsymbol{a}_n$. We deduce that u can be decomposed into $u = u'\sigma'$ where $u' \in \boldsymbol{A}_I^* \boldsymbol{a}_1 \cdots \boldsymbol{A}_I^* \boldsymbol{a}_{p-1}$ and $\sigma' \in \boldsymbol{A}_I^* \boldsymbol{a}_p \boldsymbol{A}_I^* \cdots \boldsymbol{a}_n \boldsymbol{A}_I^*$. Observe that $\Delta(\sigma')(i) = \Delta(\sigma)(i)$ for every $i \notin I$. Thus $\sigma' \in \boldsymbol{A}_{I'}^*$. From $\boldsymbol{A}_I^* \subseteq \boldsymbol{A}_{I'}^*$ we deduce that $u \in \boldsymbol{A}_{I'}^* \boldsymbol{a}_1 \boldsymbol{A}_{I'}^* \cdots \boldsymbol{a}_{p-1} \boldsymbol{A}_{I'}^*$.

Lemma VII.3 shows that there exists a sequence $n_1, \ldots, n_k \in \mathbb{N}$ such that $\boldsymbol{v}_j = \Delta(\sigma_1^{n_1} \cdots \sigma_j^{n_j})$ is a vector in $\mathbb{N}^d$ and such that $\|\boldsymbol{v}_k\|^+ = I$. Since $(\sigma_1, \ldots, \sigma_k)$ is insertable in u, there exists a decomposition of u into $u = u_1 \cdots u_{k+1}$ and a sequence $m_1, \ldots, m_k \in \mathbb{N}_{>0}$ such that $u_1 \sigma_1^{m_1} \cdots u_k \sigma_k^{m_k} u_{k+1}$ is a trace from $\boldsymbol{c}_{\text{init}}$. By monotonicity, observe that for every $r \in \mathbb{N}$ there exists a configuration $\boldsymbol{c}_r$ such that:

$$\boldsymbol{c}_{\text{init}} \xrightarrow{u_1 \sigma_1^{m_1 + r n_1} \cdots u_k \sigma_k^{m_k + r n_k} u_{k+1}} \boldsymbol{c}_r$$

Let us consider the configuration $\boldsymbol{c}'$ such that $\boldsymbol{c}_{\text{init}} \xrightarrow{u'} \boldsymbol{c}'$.

Let us prove that $\boldsymbol{c}_r(i) \geq \boldsymbol{c}'(i)$ for every $i \notin I$. Let $i \notin I$. We introduce the configuration $\boldsymbol{c}$ such that $\boldsymbol{c}' \xrightarrow{\sigma'} \boldsymbol{c}$. Note that $\boldsymbol{c}_r(i) = \boldsymbol{c}(i)$ since $u_j \in \boldsymbol{A}_I^*$. From $\boldsymbol{c} = \boldsymbol{c}' + \Delta(\sigma')$ and $\Delta(\sigma')(i) = \Delta(\sigma)(i)$, we derive $\boldsymbol{c}(i) = \boldsymbol{c}'(i) + \Delta(\sigma)(i)$. As $\|\Delta(\sigma)\|^- \subseteq I$ we get $\boldsymbol{c}(i) \geq \boldsymbol{c}'(i)$. Thus $\boldsymbol{c}_r(i) \geq \boldsymbol{c}'(i)$ for every $i \notin I$.

Let us consider $r \in \mathbb{N}$ large enough such that $r \geq \boldsymbol{c}'(i)$ for every $i \in I$. As $\boldsymbol{c}_r(i) \geq r$ for every $i \in I$ we get $\boldsymbol{c}_r \geq \boldsymbol{c}'$. Since σ' is a trace from $\boldsymbol{c}'$ we deduce that σ' is also a trace from $\boldsymbol{c}_r$. Lemma VII.4 shows that there exists $r' \in \mathbb{N}$ such that σ is a trace from $\boldsymbol{c}_r + r' \boldsymbol{v}_k$. We deduce that $u_0 \sigma_1^{m_1 + r + r'} u_1 \cdots \sigma_k^{m_k + r + r'} u_k \sigma$ is a trace from $\boldsymbol{c}_{\text{init}}$. Thus $(\sigma_1, \ldots, \sigma_k, \sigma)$ is insertable in u. ∎

The following lemma shows that a "partial witness" of non-context-freeness can be obtained from any run of $\mathcal{P}_W$. This "partial witness" is inductive and it will provide a witness of non-context-freeness when the target configuration can execute a "fail with empty stack" transition. The proof is obtain with an immediate induction over the length of the run. The nested relation R introduced in this lemma corresponds intuitively to the matching push/pop operations performed along the run. The content of the stack is obtained from unmatched elements corresponding to *free indexes* of R. An index $j \in \{1, \ldots, k\}$ is said to be *free* for a nested relation R over $\{1, \ldots, k\}$ if there does not exist $(s, t) \in R$ satisfying $s < j < t$.

Lemma VII.6. *For every configuration $(\boldsymbol{p}, w, \boldsymbol{r}, z)$ reachable in $\mathcal{P}_W$ by some run, there exists a matching scheme R for the sequence $(\sigma_1, \ldots, \sigma_k)$ of cycles extracted along the run and a sequence $j_1 < \cdots < j_m$ of free indexes for R such that:*

- *R satisfies condition (5) of Definition V.1,*
- *$\|\boldsymbol{r}\|^+ = \|\mathsf{exc}(R)\|^+$,*
- *The set of words $(\mathbb{Q}_{>0} \Delta(\sigma_{j_1})) \cdots (\mathbb{Q}_{>0} \Delta(\sigma_{j_m}))$ contains z if $\boldsymbol{p} = \boldsymbol{0}$ and it contains $z\boldsymbol{p}$ otherwise, and*
- *$j_m = k$ if $\boldsymbol{p} \not\geq \boldsymbol{0}$.*

When the state $\lightning$ is reachable in $\mathcal{P}_W$, a witness of non-context-freeness can be derived from the "partial witness" introduced in the previous lemma.

Proposition VII.7. *If the state $\lightning$ is reachable in $\mathcal{P}_W$ then $\langle \boldsymbol{A}, \boldsymbol{c}_{init} \rangle$ admits a witness of non-context-freeness.*

Proof: There exists a configuration $(\boldsymbol{p}, w, \boldsymbol{r}, z)$ reachable in $\mathcal{P}_W$ by a run labeled by u such that $\boldsymbol{p} \not\geq \boldsymbol{0}$ and $\Delta(z) = \boldsymbol{0}$. Since the vector $\boldsymbol{p} + \boldsymbol{z} + \Delta(z)$ is constant during the "non-positive processing" and it is in $\mathbb{Q}_{\geq 0}^d$ just after an "extract cycle" transition, we deduce that $\boldsymbol{p} + \boldsymbol{r} + \Delta(z) \geq \boldsymbol{0}$. Hence from $\Delta(z) = \boldsymbol{0}$, we get $\|\boldsymbol{p}\|^- \subseteq \|\boldsymbol{r}\|^+$. Lemma VII.5 shows

that the sequence $(\sigma_1, \ldots, \sigma_k)$ of cycles extracted along the run is insertable in U. Lemma VII.6 shows that there exists a matching scheme R for $(\sigma_1, \ldots, \sigma_k)$ satisfying condition (5) of Definition V.1 and such that $\|\boldsymbol{r}\|^+ = \|\mathsf{exc}(R)\|^+$ and $\boldsymbol{p} \in \mathbb{Q}_{>0}\Delta(\sigma_k)$.

There exists a decomposition of u into $u = u_1 \cdots u_{k+1}$ and a sequence $m_1, \ldots, m_k \in \mathbb{N}_{>0}$ such that $u_1\sigma_1^{m_1} \cdots u_k\sigma_k^{m_k}u_{k+1}$ is a trace from $\boldsymbol{c}_{\text{init}}$. We consider the tuple $(u_1, \sigma_1^{m_1}, \ldots, u_k, \sigma_k^{m_k}, U)$ where U is the matching relation $U = \{(2s, 2t) \mid (s,t) \in R\}$. Just observe that this tuple is a witness of non-context-freeness for $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}}\rangle$. ∎

By Proposition V.2, the trace language of a VAS that admits a witness of non-context-freeness is not context-free. We derive the following corollary.

Corollary VII.8. *If the state $\lightning$ is reachable in $\mathcal{P}_W$ then the trace language of $\langle \boldsymbol{A}, \boldsymbol{c}_{init}\rangle$ is not context-free.*

VIII. Decidability of the Context-Freeness Problem for VAS

We now show how the vector pushdown automaton $\mathcal{P}_W$, introduced in the previous section, can be used to solve the context-freeness problem for VAS. There are two possible causes for failure of $\mathcal{P}_W$. Corollary VII.8 shows that the trace language of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}}\rangle$ is not context-free when $\lightning$ is reachable in $\mathcal{P}_W$. However, reachability of $\bot$ in $\mathcal{P}_W$ only means, intuitively, that the support set W is too small. We show that there exists a finite support set that is large enough.

Proposition VIII.1. *There exists a finite support set $W \subseteq \boldsymbol{A}^*$ such that $\bot$ is not reachable in $\mathcal{P}_W$.*

Proof: Note that $\bot$ is not reachable in $\mathcal{P}_{\boldsymbol{A}^*}$. Let W be the set of all words $w \in \boldsymbol{A}^*$ such that the state $(\boldsymbol{0}, w)$ is reachable in $\mathcal{P}_{\boldsymbol{A}^*}$. It is readily seen that W is non-empty and prefix-closed. Observe that every configuration $(\boldsymbol{0}, w, \boldsymbol{r}, z)$ that is reachable in $\mathcal{P}_W$ is also reachable in $\mathcal{P}_{\boldsymbol{A}^*}$. Therefore, $\bot$ is not reachable in $\mathcal{P}_W$. It remains to show that W is finite.

Assume, by contradiction, that W is infinite. Since $\boldsymbol{A}$ is finite, we obtain by König's Lemma that there exists an infinite sequence $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots$ of actions such that $w_n = \boldsymbol{a}_1 \cdots \boldsymbol{a}_n \in W$ for every $n \in \mathbb{N}$. Let $n \in \mathbb{N}$. By definition of W, the state $(\boldsymbol{0}, w_{n+1}) = (\boldsymbol{0}, w_n\boldsymbol{a}_{n+1})$ is reachable in $\mathcal{P}_{\boldsymbol{A}^*}$. It follows that $\mathcal{P}_{\boldsymbol{A}^*}$ contains a run of the form

$$(\boldsymbol{0}, \varepsilon, \boldsymbol{0}, \varepsilon) \xrightarrow{*} (\boldsymbol{0}, w_n, \boldsymbol{r}_n, z_n) \xrightarrow{\mathsf{test}(\neg Extr(w_n))} \cdots \qquad (13)$$

Define $\boldsymbol{v}_n = \mathsf{val}(\boldsymbol{0}, w_n, \boldsymbol{r}_n, z_n)$. We derive from (12) that $\boldsymbol{v}_n \in \mathbb{N}^d$ for every $n \in \mathbb{N}$. By Dickson's Lemma, there exists indexes $m < n$ such that $\boldsymbol{v}_m \leq \boldsymbol{v}_n$. It follows that

$$\Delta(w_m) + \boldsymbol{r}_m + \Delta(z_m) \;\leq\; \Delta(w_n) + \boldsymbol{r}_n + \Delta(z_n)$$

Therefore, $\boldsymbol{r}_n + \Delta(z_n) + \Delta(\sigma) \geq \boldsymbol{0}$, where $\sigma = \boldsymbol{a}_{m+1} \cdots \boldsymbol{a}_n$. Observe that σ is a non-empty suffix of w_n. We obtain that $\boldsymbol{r}_n, \Delta(z_n) \models \mathit{Extr}(w_n)$, which contradicts (13). ∎

Even when the support set W is finite, $\mathcal{P}_W$ has infinitely many states and transitions, which is inadequate for algorithmic purposes. To address this issue, we restrict $\mathcal{P}_W$ to its reachable states and transitions. Define Q^r_W and T^r_W to be the sets of states and transitions that are reachable in $\mathcal{P}_W$, respectively. Clearly, the reduced vector pushdown automaton $\mathcal{P}^r_W = \langle Q^r_W, (\boldsymbol{0}, \varepsilon), \boldsymbol{A}, T^r_W\rangle$ has the same runs from the initial configuration as $\mathcal{P}_W$. So it recognizes the same language as $\mathcal{P}_W$. However, $\mathcal{P}^r_W$ still contains vectors of rational numbers in the state and in the stack. We first establish a sufficient condition for components of these vectors to be integers.

Lemma VIII.2. *For every configuration $(\boldsymbol{p}, w, \boldsymbol{r}, \boldsymbol{\gamma}_1 \cdots \boldsymbol{\gamma}_h)$ reachable in $\mathcal{P}_W$, the components $\boldsymbol{p}(i)$ and $\boldsymbol{\gamma}_1(i), \ldots, \boldsymbol{\gamma}_h(i)$ are integers for every index i such that $\boldsymbol{r}(i) = 0$.*

Proof: The lemma obviously holds for the initial configuration $(\boldsymbol{0}, \varepsilon, \boldsymbol{0}, \varepsilon)$. We show that the lemma condition is preserved by every step of $\mathcal{P}_W$. Consider a step $(\boldsymbol{p}, w, \boldsymbol{r}, z) \xrightarrow{t} (\boldsymbol{p}', w', \boldsymbol{r}', z')$ where t is a transition of $\mathcal{P}_W$, and assume that the lemma holds for $(\boldsymbol{p}, w, \boldsymbol{r}, z)$. We proceed by case analysis on the transition t.

If t is a "non-positive processing" transition, then it pops a vector $\boldsymbol{\gamma} \in \mathbb{Q}^d_{\geq 0}$ from the stack $\mathtt{z}$ and adds $\mathsf{rem}(\boldsymbol{\gamma}, \boldsymbol{p})$ to the counter $\mathtt{r}$. We get that $\boldsymbol{p}' = \mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{p})$, $\boldsymbol{r}' = \boldsymbol{r} + \mathsf{rem}(\boldsymbol{\gamma}, \boldsymbol{p})$, and $z = z'\boldsymbol{\gamma}$. Recall that $\boldsymbol{0} \leq \boldsymbol{r}$ and $\boldsymbol{0} \leq \mathsf{rem}(\boldsymbol{\gamma}, \boldsymbol{p})$. Consider an index $i \in \{1, \ldots, d\}$ such that $\boldsymbol{r}'(i) = 0$. We derive from $\boldsymbol{r}' = \boldsymbol{r} + \mathsf{rem}(\boldsymbol{\gamma}, \boldsymbol{p})$ that $\boldsymbol{r}(i) = \mathsf{rem}(\boldsymbol{\gamma}, \boldsymbol{p})(i) = 0$. According to the induction hypothesis, $\boldsymbol{p}(i)$ and $\boldsymbol{\gamma}(i)$ are both integers. Recall that $\boldsymbol{\gamma} + \boldsymbol{p} = \mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{p}) + \mathsf{rem}(\boldsymbol{\gamma}, \boldsymbol{p})$. It follows that $\boldsymbol{p}'(i) = \boldsymbol{\gamma}(i) + \boldsymbol{p}(i)$ is an integer. Moreover, as z' is a prefix of z, we obtain that the lemma holds for $(\boldsymbol{p}', w', \boldsymbol{r}', z')$.

The other cases for t immediately follow from the induction hypothesis. ∎

Proposition VIII.3. *For every finite support set W, the sets Q^r_W and T^r_W are finite and computable from W and $\langle \boldsymbol{A}, \boldsymbol{c}_{init}\rangle$.*

Proof: We introduce two sets, $\boldsymbol{\Gamma}_W \subseteq \mathbb{Q}^d_{\geq 0}$ and $\boldsymbol{M}_W \subseteq \mathbb{Q}^d$, and show that they contain the sets of reachable stack symbols and of reachable vectors $\boldsymbol{p}$, respectively. Firstly, $\boldsymbol{\Gamma}_W$ is the set of all vectors $\frac{k}{\Delta(\sigma)(i)}\Delta(\sigma)$ such that

- σ is a suffix of some word in W verifying $\Delta(\sigma) \geq \boldsymbol{0}$,
- i is an index in $\{1, \ldots, d\}$ with $\Delta(\sigma)(i) > 0$, and
- $k \in \{1, \ldots, \Delta(\sigma)(i)\}$.

Secondly, $\boldsymbol{M}_W$ is the least subset of $\mathbb{Q}^d$ satisfying the two following conditions:

- $\Delta(\sigma) \in \boldsymbol{M}_W$ for every $w \in W$ and every suffix σ of w,
- $\mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{m}) \in \boldsymbol{M}_W$ for every $\boldsymbol{\gamma} \in \boldsymbol{\Gamma}_W$ and $\boldsymbol{m} \in \boldsymbol{M}_W$ with $\boldsymbol{m} \not\geq \boldsymbol{0}$.

It is readily seen that $\boldsymbol{\Gamma}_W$ is finite and computable. Hence, there exists $\eta > 0$ such that $\eta \leq \boldsymbol{\gamma}(i)$ for all $\boldsymbol{\gamma} \in \boldsymbol{\Gamma}_W$ and $i \in \|\boldsymbol{\gamma}\|^+$. Observe that for every $\boldsymbol{\gamma} \in \boldsymbol{\Gamma}_W$ and $\boldsymbol{m} \in \mathbb{Q}^d$ with $\boldsymbol{m} \not\geq \boldsymbol{0}$, if $\boldsymbol{m} \neq \mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{m})$ and $\mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{m}) \not\geq \boldsymbol{0}$ then $\boldsymbol{m}(i) \leq \mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{m})(i)$ for all $i \in \|\boldsymbol{m}\|^-$ and $\boldsymbol{m}(j) + \eta \leq \mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{m})(j)$ for some $j \in \|\boldsymbol{m}\|^-$. It follows that $\boldsymbol{M}_W$ is also finite, and, hence, it is computable.

Let us show that $\boldsymbol{p} \in \boldsymbol{M}_W$ and $z \in \boldsymbol{\Gamma}^*_W$ for every configuration $(\boldsymbol{p}, w, \boldsymbol{r}, z)$ that is reachable in $\mathcal{P}_W$. The proof is by induction on the length of runs in $\mathcal{P}_W$ from its initial

$\mathsf{ContextFree}\,(\boldsymbol{A}, \boldsymbol{c}_{\text{init}})$

1 **foreach** finite support set $W \subseteq \boldsymbol{A}^*$ **do**
2 **if** $\bot$ is not reachable in $\mathcal{P}_W$ **then**
3 **if** $\lightning$ is reachable in $\mathcal{P}_W$ **then**
4 **return** `no`
5 **else**
6 **return** `yes`

Figure 3. Algorithm solving the context-freeness problem for VAS.

configuration. The initial configuration $(\mathbf{0}, \varepsilon, \mathbf{0}, \varepsilon)$ obviously satisfies the desired property. Consider a run

$$(\mathbf{0}, \varepsilon, \mathbf{0}, \varepsilon) \xrightarrow{*} (\boldsymbol{p}, w, \boldsymbol{r}, z) \xrightarrow{t} (\boldsymbol{p}', w', \boldsymbol{r}', z')$$

where t is a transition of $\mathcal{P}_W$, and assume that $\boldsymbol{p} \in \boldsymbol{M}_W$ and $z \in \boldsymbol{\Gamma}_W^*$. A routine inspection of $\mathcal{P}_W$'s transition rules shows that $\boldsymbol{p}' \in \boldsymbol{M}_W$ and $z' \in \boldsymbol{\Gamma}_W^*$. We detail the non-trivial cases.

If t is a "non-positive processing" transition, then $\boldsymbol{p} \not\geq \mathbf{0}$ and the vector $\boldsymbol{\gamma}$ that is popped from the stack satisfies $z = z'\boldsymbol{\gamma}$. Since $z \in \boldsymbol{\Gamma}_W^*$, we get that $\boldsymbol{\gamma} \in \boldsymbol{\Gamma}_W$. It follows from the definition of $\boldsymbol{M}_W$ that $\boldsymbol{p}' = \mathsf{mat}(\boldsymbol{\gamma}, \boldsymbol{p}) \in \boldsymbol{M}_W$.

If t is a "positive processing" transition and $\|\boldsymbol{p}\|^+ \not\subseteq \|\boldsymbol{r}\|^+$, then $\boldsymbol{r}(i) = 0 < \boldsymbol{p}(i)$ for some index $i \in \{1, \ldots, d\}$. It follows from Lemma VIII.2 that $\boldsymbol{p}(i) \in \mathbb{N}$. Moreover, since $\boldsymbol{p} \in \boldsymbol{M}_W$ and $\boldsymbol{p} > \mathbf{0}$, we obtain that $\boldsymbol{p} = \mu\Delta(\sigma)$ where σ is a suffix of some word in W verifying $\Delta(\sigma) \geq \mathbf{0}$, and μ is a rational number such that $0 < \mu \leq 1$. Observe that $\Delta(\sigma)(i) > 0$ and $\mu = \frac{\boldsymbol{p}(i)}{\Delta(\sigma)(i)}$. This entails that $\boldsymbol{p} \in \boldsymbol{\Gamma}_W$, hence, $z' = z\boldsymbol{p}$ is contained in $\boldsymbol{\Gamma}_W^*$.

We have shown that $\boldsymbol{p} \in \boldsymbol{M}_W$ for every state $(\boldsymbol{p}, w)$ in Q_W^r, and $\boldsymbol{\gamma} \in \boldsymbol{\Gamma}_W$ for every $\mathsf{push}(\boldsymbol{\gamma})$ or $\mathsf{pop}(\boldsymbol{\gamma})$ operation of T_W^r. Therefore, by restricting $\boldsymbol{\gamma}$ and $\boldsymbol{p}$ to $\boldsymbol{\Gamma}_W$ and $\boldsymbol{M}_W$, respectively, we obtain a computable finite set $Q_W^\sharp$ of states of $\mathcal{P}_W$ and a computable finite set $T_W^\sharp$ of transitions of $\mathcal{P}_W$ such that $Q_W^r \subseteq Q_W^\sharp$ and $T_W^r \subseteq T_W^\sharp$. Thus, the sets Q_W^r and T_W^r are both finite. Clearly, the vector pushdown automaton $\mathcal{P}_W^\sharp = \langle Q_W^\sharp, (\mathbf{0}, \varepsilon), \boldsymbol{A}, T_W^\sharp \rangle$ has the same runs from the initial configuration as $\mathcal{P}_W$. We derive from Proposition VI.1 that the finite sets Q_W^r and T_W^r are computable. ∎

We are now equipped with the required ingredients to present our algorithm solving the context-freeness problem for VAS (see Figure 3). The algorithm looks for a finite support set W that is large enough, i.e., such that $\bot$ is not reachable in $\mathcal{P}_W$. Then the algorithm returns `yes` if $\mathcal{P}_W$ succeeds, and `no` otherwise. Proposition VIII.3 guarantees that the tests performed at lines 2 and 3 are computable. Termination of this algorithm follows from Proposition VIII.1. Its correctness derives from Corollary VII.8 for line 4, and from Propositions VI.1 and VII.2 for line 6. This concludes our proof that the context-freeness problem for VAS is decidable.

Remark VIII.4. When $\mathsf{ContextFree}\,(\boldsymbol{A}, \boldsymbol{c}_{\text{init}})$ returns `yes`, a finite support set $W \subseteq \boldsymbol{A}^*$ such that $\mathcal{P}_W$ succeeds has been computed by the algorithm. According to Proposition VIII.3, the reduced vector pushdown automaton $\mathcal{P}_W^r$ has finitely many states and transitions, and is computable. We derive from Proposition VI.1 that the trace language of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ is effectively context-free.

IX. Conclusion

When the trace language of a VAS $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ is not context-free, the state $\lightning$ is reachable in $\mathcal{P}_W$, where $W = \boldsymbol{A}^*$. Proposition VII.7 shows that $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ admits a witness of non-context-freeness. We derive from Proposition V.2 that the intersection of the trace language of $\langle \boldsymbol{A}, \boldsymbol{c}_{\text{init}} \rangle$ with a bounded regular language is not context-free. Since context-free languages are closed under intersection with regular languages, we obtain the following characterization, which cannot be derived from Schwer's proof (see [9, p. 224]).

> *The trace language of a VAS is context-free if, and only if, it has a context-free intersection with every bounded regular language.*

We conjecture, based on this characterization, that the context-freeness problem for VAS is solvable in exponential space.

Acknowledgment

The authors thank Alain Finkel for fruitful discussions. This work was supported by the ANR project ReachHard (ANR-11-BS02-001).

References

[1] J. Esparza and M. Nielsen, "Decidability issues for petri nets - a survey," *Bulletin of the EATCS*, vol. 52, pp. 244–262, 1994.
[2] R. M. Karp and R. E. Miller, "Parallel program schemata," *J. Comput. Syst. Sci.*, vol. 3, no. 2, pp. 147–195, 1969.
[3] R. Valk and G. Vidal-Naquet, "Petri nets and regular languages," *J. Comput. Syst. Sci.*, vol. 23, no. 3, pp. 299–325, 1981.
[4] E. W. Mayr and A. R. Meyer, "The complexity of the finite containment problem for petri nets," *J. ACM*, vol. 28, no. 3, pp. 561–576, 1981.
[5] C. Rackoff, "The covering and boundedness problems for vector addition systems," *Theoretical Comput. Sci.*, vol. 6, no. 2, pp. 223–231, 1978.
[6] M. F. Atig and P. Habermehl, "On yen's path logic for petri nets," *Int. J. Found. Comput. Sci.*, vol. 22, no. 4, pp. 783–799, 2011.
[7] M. Blockelet and S. Schmitz, "Model checking coverability graphs of vector addition systems," in *Proc. MFCS'11*, ser. LNCS, vol. 6907. Springer, 2011, pp. 108–119.
[8] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981.
[9] S. R. Schwer, "The context-freeness of the languages associated with vector addition systems is decidable," *Theoretical Comput. Sci.*, vol. 98, no. 2, pp. 199–247, 1992.
[10] H.-C. Yen, "A note on fine covers and iterable factors of vas languages," *Inf. Process. Lett.*, vol. 56, no. 5, pp. 237–243, 1995.
[11] S. Ginsburg, *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, 1966.
[12] A. Ginzburg and M. Yoeli, "Vector addition systems and regular languages," *J. Comput. Syst. Sci.*, vol. 20, no. 3, pp. 277–284, 1980.
[13] S. Demri, "On selective unboundedness of vass," in *Proc. INFINITY'10*, ser. EPTCS, vol. 39, 2010, pp. 1–15.
[14] P. Ganty, R. Majumdar, and B. Monmege, "Bounded underapproximations," in *Proc. CAV'10*, ser. LNCS, vol. 6174. Springer, 2010, pp. 600–614.
[15] J. Esparza, P. Ganty, and R. Majumdar, "A perfect model for bounded verification," in *Proc. LICS'12*. IEEE Computer Society, 2012, pp. 285–294.
[16] P. Jancar, J. Esparza, and F. Moller, "Petri nets and regular processes," *J. Comput. Syst. Sci.*, vol. 59, no. 3, pp. 476–503, 1999.
[17] S. Ginsburg and E. Spanier, "Semigroups, Presburger formulas and languages," *Pacific J. Math.*, vol. 16, no. 2, pp. 285–296, 1966.