On Continuous Pushdown VASS in One Dimension

Guillermo A. Pérez
University of Antwerp – Flanders Make, Antwerp, Belgium

Shrisha Rao
University of Antwerp – Flanders Make, Antwerp, Belgium

Abstract

A pushdown vector addition system with states (PVASS) extends the model of vector addition systems with a pushdown stack. The algorithmic analysis of PVASS has applications such as static analysis of recursive programs manipulating integer variables. Unfortunately, reachability analysis, even for one-dimensional PVASS is not known to be decidable. So, we relax the model of one-dimensional PVASS to make the counter updates continuous and show that in this case reachability, coverability, and boundedness are decidable in polynomial time. In addition, for the extension of the model with lower-bound guards on the states, we show that coverability and reachability are NP-complete, and boundedness is coNP-complete.

2012 ACM Subject Classification Theory of computation → Grammars and context-free languages; Theory of computation → Concurrency

Keywords and phrases Vector addition systems, Pushdown automata, Reachability

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2024.34

Funding Work supported by the Flemish inter-university (iBOF) "DESCARTES" project.

Acknowledgements We thank Georg Zetzsche for help with the hardness proofs in the model with lower bounds; A. R. Balasubramanian for his comments on an early version of this work, and Tim Leys and Ritam Raha for useful discussions on the topic of (continuous) counter automata.

1 Introduction

Vector addition systems with states (VASS) are commonly used to model distributed systems and concurrent systems with integer variables. A VASS consists of a set of (control) states and a set of counters. Transitions between states are labelled with vectors of integers (usually encoded in binary) that are added to the current values of the counters. Importantly, transitions resulting in a counter value becoming negative are disallowed.

An equivalent way of understanding this model is to see the counters as unary-alphabet stacks. This alternative formulation has a natural extension obtained by adding one general stack to it. Pushdown VASS (PVASS), as they are usually called, can be used to model recursive programs manipulating integer variables [17, Sec. 6.2]. Arguably the most basic question one can attempt to answer algorithmically in a computational model is that of *reachability*. In the context of (pushdown) VASS, we ask whether a given target configuration (formed by the current state and the values of the counters) can be seen along a run from a given source configuration. While the complexity of reachability for VASS is now better understood [15, 6], it is not known to be decidable for PVASS and the best known lower bound is HyperAck-hardness [14]. The problem is not known to be decidable even for one dimension and the known lower bound is Pspace-hardness [7].

Motivated by the (complexity) gap in our understanding of reachability for PVASS, researchers have studied the problem for different relaxations of the model: A PVASS is *bidirected* [9] if the effect (on the stack and counters) of every transition can be (immediately) reversed; A $\mathbb{Z}$-PVASS [11] allows counters to hold negative values; A *continuous* PVASS [2] instead allows them to hold nonnegative values and counter updates labelling a transition

Table 1 Previously known complexity bounds (in black) and our bounds (in green) for problems in PVASS and relaxations thereof.

		PVASS	B-PVASS	$\mathbb{Z}$-PVASS	CPVASS	lb-CPVASS
Gen.	Reach	HypAck-h	$\in$Ack/Tower-h [9]	NP-comp.	NEXP-c	Undec. [1]
1-dim.	Reach	Pspace-h	$\in$Pspace	NP-comp.	Ptime-c	NP-c
	Cover	$\in$EXPspace	$\in$Pspace [9]	$\in$NP	Ptime-c	NP-c
	Bound	$\in$HypAck	$\in$HypAck	$\in$HypAck	Ptime-c	coNP-c

can be scaled by any $\gamma \in (0, 1]$ when taking the transition. For all of these, reachability is known to be decidable. For some of them, lower complexity bounds for the special case of one dimension have also been established. See Table 1 for a summary of known results.

In this work, we study reachability, coverability, and boundedness for continuous PVASS in one dimension. The boundedness problem asks whether the set of all reachable configurations, from a given source configuration, is bounded. In turn, coverability asks whether a vector at least as large as the given vector can be reached. In contrast to reachability, coverability is known to be decidable and in EXPspace for PVASS in one dimension [16]. Similarly, boundedness is known to be decidable and in HyperAck, this time in general, not only in one dimension [14].

Contributions

In this paper, we prove that, for continuous PVASS in one dimension, reachability, coverability, boundedness and even computing the infimum bound are Ptime-complete. We further show that if one adds to the model lower-bound guards on the states for the counter (thus allowing for a "tighter" relaxation of the original model, since one can now partially control the counter values before a transition), then reachability and coverability are NP-complete while boundedness is coNP-complete.

2 Preliminaries

We first recall a definition of pushdown automata. Then, we extend it to continuous PVASS.

2.1 Pushdown automata and Context-free grammars

▶ **Definition 1** (Pushdown automata). *A pushdown automaton (PDA, for short) is a tuple* $\mathcal{P} = (S, \Sigma, \Gamma, \delta, s_0, \bot, F)$ *where* S *is a finite set of states,* Σ *a finite (possibly empty) alphabet,* Γ *a finite stack alphabet,* $s_0 \in S$ *the initial state,* $\bot \in \Gamma$ *the initial stack symbol,* $F \subseteq S$ *a set of accepting states, and* $\delta : S \times S \to \big(\Sigma \cup \epsilon\big) \times \big(\{a, \overline{a} \mid a \in \Gamma \backslash \bot\} \cup \epsilon\big)$ *a partial function, where,* a *and* $\overline{a}$ *denote pushing and popping* a *from the stack respectively.*

A *configuration* of a PDA $\mathcal{P}$ is of the form $(s, w, \alpha) \in S \times \Sigma^* \times \Gamma^*$ where s represents the current state of the PDA, w the word read by the PDA until reaching the state s and α the current stack contents of the PDA (with the right being the "top" from which we pop and onto which we push). The *initial configuration* q_0 is $(s_0, \epsilon, \bot)$.

A *run* of a PDA $\mathcal{P}$ is of the form $\pi = q_0 q_1 \dots q_n$ where the $q_i = (s_i, w_i, \alpha_i)$ are configurations and for all $0 \leq i < n$, $\delta(s_i, s_{i+1})$ is defined, $w_{i+1} = w_i \cdot \delta(s_i, s_{i+1})_1$, $\alpha_{i+1} = \alpha_i$ if $\delta(s_i, s_{i+1})_2 = \epsilon$, $\alpha_{i+1} = \alpha_i \cdot a$ if $\delta(s_i, s_{i+1})_2 = a$, and $\alpha_i = \alpha_{i+1} \cdot a$ if $\delta(s_i, s_{i+1})_2 = \overline{a}$. Above, $\delta(\_, \_)_i$ represents the i-th component. For any $Q \subseteq S$, we say the run reaches Q if $s_n \in Q$.

We focus on state reachability, that is, a run π of a PDA is *accepting* if q_0 is the initial configuration and $s_n \in F$.

The language of a PDA $\mathcal{P}$, denoted by $L(\mathcal{P})$, is the set of all words $w_n \in \Sigma^*$ read by accepting runs $q_0 \dots (s_n, w_n, \alpha_n)$ of $\mathcal{P}$. The *Parikh image* $\Phi(w)$ of a word $w \in \Sigma^*$, i.e. the vector in $\mathbb{N}^{|\Sigma|}$ such that its i^{th} component is the number of times the i^{th} letter of Σ (assuming an arbitrary choice of total order) appears in w.

Context-free grammars

CFGs, for short, are a model that is expressively equivalent to PDAs in terms of their languages. The models are logspace reducible to each other [13, Section 5.3].

▶ **Definition 2** (Context-free grammars). *A CFG is a tuple $G = (V, \Sigma, P, S)$, where V is a set of variables; Σ, a set of terminals; $P \subset V \times \{V, \Sigma\}^*$, a set of productions; and $S \in V$, the start symbol.*

A production $(A, w) \in P$ is written as $A \to w$, where the *production symbol* "$\to$" separates the *head* (a variable) of the production, to the left of $\to$, from the *body* (a string of variables and terminals) of the production, to the right of $\to$. Each variable represents a language, i.e., a (possibly empty) set of strings of terminals. The body of each production represents one way to form strings in the language of the head.

▶ **Example 3.** The grammar $G = (\{A\}, \{a, b\}, P, S = A)$ represents the set of all palindromes over $\{a, b\}$ where the productions are $A \to \epsilon, A \to a, A \to b, A \to aAa$ and $A \to bAb$. The word $abaaaba$, for example, is in the language of A since it can be obtained by $A \to aAa \to abAba \to abaAaba \to abaaaba$ where the fourth, fifth, again the fourth, and finally, the second production rules are applied, in that order.

Chomsky normal form (or CNF) [13, Section 4.5] is a normal form for CFGs with the restriction that all production rules can only be of the form $A \to BC$, or $A \to a$, or $S \to \varepsilon$. Every CFG has an expressively equivalent CFG in CNF.

2.2 Continuous pushdown VASS

A CPVASS in one dimension is a PDA with a continuous counter.

▶ **Definition 4** (lb-C1PVASS). *A continuous pushdown VASS (with lower-bound guards) in one dimension is a tuple $\mathcal{A} = (S, \Sigma, \Gamma, \delta, \bot, s_0, F, \ell)$ where S is a finite set of states; $s_0 \in S$, the initial state; $F \subseteq S$, a set of accepting states; Σ, a finite alphabet; Γ, a finite stack alphabet; $\bot \in \Gamma$, the initial stack symbol; $\delta : S \times S \to \big(\Sigma \cup \epsilon\big) \times \mathbb{Z} \times \big(\{a, \overline{a} \mid a \in \Gamma \backslash \bot\} \cup \epsilon\big)$, a partial transition function; and $\ell : S \to \mathbb{N}$, a function that assigns lower bounds to states.*

Since we only study runs, and not languages, of lb-C1PVASS, we henceforth omit Σ. We also assume, without loss of generality, that the set F is a singleton. This can be done by adding a new final state f' to S and adding transitions for all $f \in F$ to f' which read ϵ, have a $+0$ counter update, and do not modify the stack. With these assumptions, we have a simpler representation of a lb-C1PVASS $\mathcal{A} = (S, \Gamma, \delta, \bot, s_0, f, \ell)$ where δ is now of the form $\delta : S \times S \to \mathbb{Z} \times \big(\{a, \overline{a} \mid a \in \Gamma \backslash \bot\} \cup \epsilon\big)$.

A *configuration* of a lb-C1PVASS is of the form (s, α, c) where s and α are as for PDAs, and $c \in \mathbb{R}_{\geq 0}$ is the current nonnegative value of the counter with the property that $c \geq \ell(s)$, that is, the counter value at a state must be at least the lower bound on that state. The *initial configuration* q_0 of the lb-C1PVASS is $(s_0, \bot, 0)$.

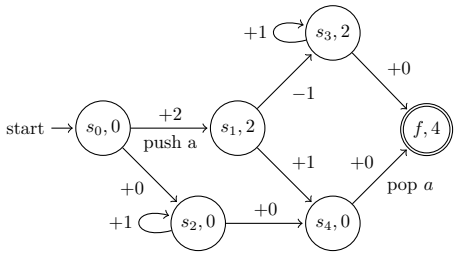

Figure 1 An example of a lb-C1PVASS $\mathcal{A}$.

A *run* of the lb-C1PVASS $\mathcal{A}$ is a sequence of configurations $\pi = q_0 q_1 \dots q_n$ with $q_i = (s_i, \alpha_i, c_i)$ such that $\pi|_{\mathcal{P}_\mathcal{A}}$, obtained by removing the counter values c_i, is a run in the PDA $\mathcal{P}_\mathcal{A}$, which is obtained by removing the counter updates from $\mathcal{A}$, and for all $0 \leq i < n$, $c_{i+1} = c_i + \gamma\delta(s_i, s_{i+1})_1$ holds for some $\gamma \in \mathbb{R} \cap (0, 1]$ where γ are the *scaling factors*.

▶ **Example 5.** Figure 1 shows a lb-C1PVASS with 6 states. The second component of the tuple inside the states denotes the lower bound on that state. For instance, $\ell(s_1) = 2$. This lb-C1PVASS does not have any run reaching f. This is because the only way to make the counter reach 4 is via s_2 or s_3. The run through s_2 does not push an a into the stack which has to be popped later in order to reach f. Also, s_3 cannot be reached, since there are only two updates $+2$ and -1 before s_3 and $\gamma_1 \cdot 2 + \gamma_2 \cdot (-1) < 2$ for all $\gamma_1, \gamma_2 \in (0, 1]$.

Acceptance conditions of a lb-C1PVASS

There are two classical ways of extending (state reachability) acceptance from runs of a PDA to runs $\pi = q_0 \dots q_n$ of lb-C1PVASS, namely: *reachability* and *coverability* for $k \in \mathbb{R}_{\geq 0}$.
***k*-Reachability** says the run is accepting if $\pi|_{P_\mathcal{A}}$ is accepting in $P_\mathcal{A}$ and $c_n = k$.
***k*-Coverability** says the run is accepting if $\pi|_{P_\mathcal{A}}$ is accepting in $P_\mathcal{A}$ and $c_n \geq k$.
Like in PDAs, $q_0 = (s_0, \alpha_0, c_0) = (s_0, \bot, 0)$ means that accepting runs start with the initial configuration. We refer to accepting runs according to the above conditions as k-reaching and k-covering runs, respectively.

We make the following simplifying assumption. All the counter updates in the transition function are in the set $\{-1, +0, +1\}$. This is no loss of generality, due to the following lemma.

▶ **Lemma 6.** *Given a lb-C1PVASS $\mathcal{A} = (S, \Gamma, \delta, \bot, s_0, f, \ell)$, there exists an equivalent lb-C1PVASS*[1] *with counter updates in the set $\{-1, +0, +1\}$, which is quadratic in the size of the encoding of $\mathcal{A}$, thus polynomial even if the counter updates are encoded in binary.*

The proof follows from the construction of a simple gadget that takes as input the binary encoding of the update and outputs that exact number of $+1$ (or -1) updates.

We also study lb-C1PVASS where all lower-bound guards are 0 (a looser approximation of PVASS) where we give simpler algorithms to solve the decision problems we consider.

▶ **Definition 7** (C1PVASS). *A C1PVASS is a lb-C1PVASS $\mathcal{A} = (S, \Gamma, \delta, \bot, s_0, f)$ where $\ell(s) = 0$ for all $s \in S$.*

For C1PVASS, we omit ℓ. *Configurations*, *runs*, and *accepting runs* are defined similarly to lb-C1PVASS. Note that, in a configuration (s, α, c), instead of $c \geq \ell(s)$, we now only have the restriction that $c \geq 0$, that is, the counter values never go below 0.

[1] To be precise: there is a clear relation between their sets of reachable configurations.

▶ **Example 8.** In Figure 1, if all the lower bounds were 0, then we would be able to reach f with a counter value of at least 4 by taking the run to s_3 and taking the self loop a few times before entering f with a counter value of at least 4. However, the run via s_2 would still not be a 4-reaching run since there is no a to pop from the stack when the run reaches s_4.

2.3 Decision problems

We focus on the computational complexity of two decision problems we call reachability and coverability, respectively: Given a lb-C1PVASS and $k \in \mathbb{N}$ (in binary), determine whether it has a k-reaching run. Given a lb-C1PVASS and $k \in \mathbb{N}$ (in binary), determine whether it has a k-covering run. In addition, we also study the complexity of the boundedness problem: Given a lb-C1PVASS, determine whether for some $k \in \mathbb{R}_{\geq 0}$ it has no k-covering run.

▶ Remark 9. Note that our definition of lb-C1PVASS is equivalent to the one where δ and ℓ map to rationals and $k \in \mathbb{Q}_{\geq 0}$, that is, $\delta : S \times S \to \big(\Sigma \cup \epsilon\big) \times \mathbb{Q} \times \big(\{a, \overline{a} \mid a \in \Gamma \backslash \bot\} \cup \epsilon\big)$ and $\ell : S \to \mathbb{Q}_{\geq 0}$, as one can multiply all counter updates, lower bounds and k by the product of all the denominators. Since the numbers are encoded in binary, the representation of the new integers will be polynomial in the size of the rationals (i.e. the bitsize of integer pairs). This preserves all the properties studied in this paper.

3 Counter properties of C1PVASS

We first show a relation between reachability and coverability.

▶ **Lemma 10.** *The reachability and coverability problems are equivalent for C1PVASS with $k > 0$, but 0-coverability does not imply 0-reachability.*

Proof. By definition, k-reachability implies k-coverability. To show the converse, take any covering run with counter value at the end of the run being $k + c$, for some $c \geq 0$. Now, we modify the run by scaling all of the counter updates in that run by $\frac{k}{k+c}$. The reader can easily verify that this is indeed a reaching run.

This proof does not work for $k = 0$ since we cannot scale the counter updates by 0. A simple example for the second part of the lemma would be a C1PVASS with a single transition, which goes from s_0 to f with a $+1$ counter update (and no stack update). In this case, 0 can be covered but not reached. ◀

From the proof above we directly get the following.

▶ Remark 11. Let $k \in \mathbb{N}_{>0}$. Then, for all $k' \in (0, k]$, k-reachability implies k'-reachability.

We also have the following simple observation about the first nonzero counter update due to our choice of q_0.

▶ Remark 12. Along any run, the first nonzero counter update must be positive, since the updates cannot be scaled to 0 and the counter values must always be nonnegative

Since we have shown that k-reachability and coverability are different for $k = 0$ but the same for $k > 0$, we first analyse the complexity of 0-reachability and 0-coverability in Section 3.1. We then show, in Section 3.2, that boundedness is decidable in PTime and that, if a C1PVASS is bounded, computing the infimum upper bound is also in PTime. Finally, in Section 3.3, we leverage the size of the upper bound along with Remark 11 to show that k-reachability and k-coverability, for $k > 0$, are also in PTime.

3.1 0-reachability and 0-coverability

In this section, reduce both 0-reachability and 0-coverability to checking nonemptiness of PDAs (with an empty alphabet) to show the following:

▶ **Theorem 13.** *The* 0*-reachability and* 0*-coverability problems for a C1PVASS are decidable in* PTime.

The result follows from the fact that checking nonemptiness of the language of a PDA can be done in polynomial time (see, e.g. [13, Proof of Lemma 4.1]) and the following lemma.

▶ **Lemma 14.** *The* 0*-reachability and* 0*-coverability problems for C1PVASS are polynomial time reducible to the nonemptiness of the language of a PDA.*

Proof sketch. For 0-coverability, first note the fact that any run in the underlying PDA of the C1PVASS $\mathcal{A}$ corresponds to a 0-covering run in $\mathcal{A}$ if and only if the first non-zero counter update in the run is a $+1$. We design a PDA as shown in Figure 2 (the blue dashed box) which simulates this property of 0-covering runs, where $\mathcal{P}_0$ and $\mathcal{P}_1$ are copies of the underlying PDA where $\mathcal{P}_0$ only has transitions corresponding to $+0$ updates in $\mathcal{A}$, and the run can enter $\mathcal{P}_1$ only after a $+1$ update.

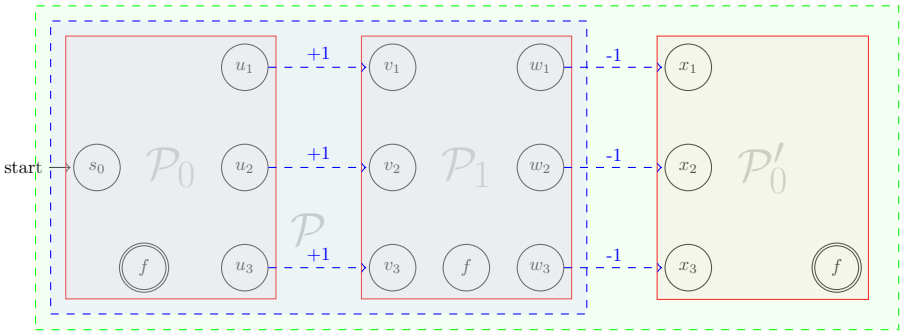

Figure 2 The construction of the PDA $\mathcal{P}$ where $\mathcal{P}_0$ and $\mathcal{P}'_0$ are copies of $\mathcal{A}$ obtained by the counter and removing all the transitions that have a nonzero counter update; the transitions from $\mathcal{P}_0$ to $\mathcal{P}_1$ are exactly the transitions in $\mathcal{A}$ with a positive counter update and those from $\mathcal{P}_1$ to $\mathcal{P}'_0$ are exactly the ones with a negative counter update. For coverability, $\mathcal{P}$ consists of just $\mathcal{P}_0$ and $\mathcal{P}_1$ and the copies of accepting states in $\mathcal{P}_1$ are also accepting in this case.

For reachability, note that a run in the underlying PDA of $\mathcal{A}$ corresponds to a 0-reaching run in $\mathcal{A}$ if and only if the first non-zero counter update is $+1$ and the last non-zero counter update is -1. Figure 2 (the green dashed box) simulates this by creating 3 copies of the underlying PDA of $\mathcal{A}$, namely, $\mathcal{P}_0$, $\mathcal{P}_1$ and $\mathcal{P}'_0$ where $\mathcal{P}_0$ and $\mathcal{P}'_0$ only contain transitions corresponding to $+0$ updates in $\mathcal{A}$. ◀

3.2 Boundedness for C1PVASS

In this section, we first analyze the complexity of deciding whether a C1PVASS is bounded or not. If it is bounded, we provide a bound which is polynomial (when encoded in binary) in the size of the encoding of the C1PVASS. We next show that, for a bounded C1PVASS, the "tight" bound, that is,

$$b = \inf\{k \in \mathbb{R} \mid \mathcal{A} \text{ has no } k\text{-covering run}\} \tag{1}$$

is an integer and the natural decision problem associated to finding b is in PTime. First, we convert the C1PVASS into a PDA $\mathcal{P}'$ as we did in the proof of Lemma 14 in Figure 2 (the blue dashed box), further modify its alphabet, and observe some properties about the resulting PDA.

Let $\mathcal{A}$ be the C1PVASS. Make two copies $\mathcal{P}_0$ and $\mathcal{P}_1$ of $\mathcal{A}$ without the counter. Next, remove from $\mathcal{P}_0$ all the transitions that were not a $+0$ counter update in $\mathcal{A}$ and add, for each transition in $\mathcal{A}$ with a positive counter update, a transition from $\mathcal{P}_0$ to $\mathcal{P}_1$. The copies of accepting states in $\mathcal{P}_0$ and $\mathcal{P}_1$ are all accepting in the resulting PDA, which we call $\mathcal{P}$ (see the blue dashed box in Figure 2). To obtain $\mathcal{P}'$ from $\mathcal{P}$, we modify its alphabet. The alphabet Σ of $\mathcal{P}'$ is unary, i.e. $\Sigma = \{a\}$. The transitions of $\mathcal{P}'$ read a if they had a $+1$ counter update in $\mathcal{A}$ and read the empty letter ϵ otherwise.

One can see a relation between accepting runs in $\mathcal{A}$ and $\mathcal{P}'$. Let π be an accepting run in $\mathcal{P}'$. The corresponding run in $\mathcal{A}$ has the property that the first nonzero update is a positive update (i.e., a transition from $\mathcal{P}_0$ to $\mathcal{P}_1$) which makes it an accepting run in $\mathcal{A}$. Similarly, an accepting run in $\mathcal{A}$ must have a $+1$ as the first nonzero update, hence, it is also an accepting run in $\mathcal{P}'$ by construction.

The lemma below follows immediately from the construction.

▶ **Lemma 15.** *$a^m \in L(\mathcal{P}')$ if and only if there is an accepting run in $\mathcal{A}$ with exactly m many $+1$ updates.*

For all $0 < \varepsilon < 1$, and an accepting run in the PDA $\mathcal{P}'$, in the corresponding run in $\mathcal{A}$, one can choose $\gamma = 1$ for all the $+1$ updates and $\gamma \in (0, 1]$ small enough, for all negative updates, so that their sum is in the interval $[0, \varepsilon)$. This leads to the following result.

▶ **Lemma 16.** *The cardinality of $L(\mathcal{P}')$ is bounded if and only if the C1PVASS $\mathcal{A}$ is bounded. Moreover, if the maximum length of a word accepted by $\mathcal{P}'$ is $p \in \mathbb{N}$ then $b = p$, where b is as in Equation* (1).

There are Ptime algorithms (see, e.g., [13, Theorem 6.6]) to determine whether the language of a PDA is finite. We thus get:

▶ **Theorem 17.** *Deciding boundedness of a C1PVASS $\mathcal{A}$ is in PTime. Moreover, the bound can be at most $2^{O(|\mathcal{A}|^6)}$, where $|\mathcal{A}|$ is the size of the encoding of $\mathcal{A}$.*

The first part of the proof follows from the discussion above. The bound is due to the facts that any PDA of size n can be converted to a CFG in CNF of size at most $O(n^6)$ (at most $2500n^6$ to be exact), the size of $\mathcal{P}$ is at most twice that of $\mathcal{A}$, and 2^m is a bound on the length of the words accepted by a (bounded) CFG in CNF of size m.

Using this upper bound on the largest reachable counter for a bounded C1PVASS, we argue the tight upper bound is an integer and give an algorithm to compute it.

▶ Remark 18. Using Lemma 15 and the fact that nonnegative updates can be scaled down arbitrarily, one can see that the bound b defined in Equation (1) is a nonnegative integer when it exists.

▶ **Theorem 19.** *The tight upper bound of a bounded C1PVASS can be computed in PTime.*

Proof. The idea for the proof comes from [8] which gives a PTime algorithm to find the shortest word accepted by a CFG.

Assume the language is not empty. Construct the PDA described in Lemma 16. We know that if m is the length of a longest word accepted by the PDA $\mathcal{P}$, then m is the tight bound. We also know, by Theorem 17, that $m \leq 2^k$ where $k = O(|\mathcal{A}|)^6$. We construct a

grammar (V, Σ, P, S) in CNF for the PDA. This grammar has size at most $2^{O(|\mathcal{A}|)^6}$, as shown in Theorem 17. Since the grammar is in CNF, all productions are of the form $A \to BC$ or $A \to a$ and the language of all variables is nonempty.

Define the function $N : V \to \mathbb{N}$ such that $N(A)$ is the length of the longest word produced by the variable A, for all $A \in V$. The following algorithm computes $N(A)$ for all $A \in V$.

1. Initialize $W(A) = 0$ for all $A \in V$, $W(a) = 1$ for all $a \in T$.
2. Repeat, for all A and all productions with head A:

$$W(A) = \begin{cases} \max\{W(B) + W(C), W(A)\} & \text{if } A \to BC; \\ \max\{W(a), W(A)\} & \text{if } A \to a. \end{cases}$$

   until we reach a fixed point (we know a fixed point will be reached eventually since the length of words is bounded).
3. Output the vector $W(V)$.

We know that the above algorithm terminates since the length of the longest word is bounded. It remains to show that it terminates in polynomially many iterations. Each iteration has $|V||P|$ comparisons of numbers bounded by $2^{O(|\mathcal{A}|)^6}$, and we know that such numbers can be compared in time polynomial in $|\mathcal{A}|$. Hence, showing that the fixed point is obtained in polynomially many iterations of the algorithm suffices to establish that the tight bound can be obtained in polynomial time.

Consider the directed graph with $V \cup \Sigma$ as vertices and where we add the edge (A, β), where $A \in V$ and $\beta \in V \cup \Sigma$, if and only if there is a production in P whose head is A and with β in its body. The graph can be shown to be acyclic, since the language of the grammar is finite and the language of every variable is nonempty. Now, every iteration of the algorithm induces a labelling of the vertices of the graph via W. Observe that the label of a vertex only changes if the label of one of its immediate successors changes. It follows that the fixed point is reached after at most $|V|$ iterations. ◀

▶ **Lemma 20.** *The set of all reachable values in a C1PVASS is closed on the right (i.e., the bound b can be reached) if and only if there is an accepting run for a^b in $\mathcal{P}'$ which does not contain any -1 transitions from $\mathcal{A}$.*

The proof follows from the simple fact that any -1 update in $\mathcal{A}$ cannot be scaled down to 0, and b is an upper bound on the counter value in the final configuration of any accepting run.

Lemma 20 gives us an easy way to check whether the interval of all reachable counter values is closed on the right. Remove all transitions from $\mathcal{P}'$ which correspond to a -1 update transition in $\mathcal{A}$. This PDA $\mathcal{P}''$ will accept a^m if and only if $\mathcal{A}$ has an accepting run with exactly m many $+1$ updates and no negative updates. This can be checked in PTime due to [5].

3.3 k-reachability and k-coverability for $k > 0$

The following stronger theorem implies that both k-reachability and coverability are in PTime for all $k \geq 0$.

▶ **Theorem 21.** *The interval of all reachable counter values of a C1PVASS is computable in polynomial time.*

Proof. Use Theorem 13 to decide whether 0 is reachable. If so, the interval is closed on the left, open otherwise. Next, use Theorem 17 to decide if the highest reachable counter value is bounded. If not, the upper bound will be ∞ (and thus open). If it is bounded, use Theorem 19 to compute the tight bound b. Finally, use Lemma 20 to find whether the interval is closed on the right. ◀

PTIME-hardness for all the problems in this section follows from the nonemptiness problem for PDAs being PTIME-hard (see, e.g. [5, Prop. 1]). For coverability and reachability this is immediate, for boundedness one can add a self loop with a positive counter update on accepting states.

4 Counter properties of lb-C1PVASS

In this section, we show that coverability and reachability for lb-C1PVASS are decidable in NP and comment on how our treatment of boundedness from the previous section adapts almost identically to lb-C1PVASS to yield, in this case, a complexity of coNP. Finally, we provide a two-step reduction from the subset-sum problem to show completeness in the respective complexity classes.

Both for coverability and reachability, we proceed as follows. First, we convert the given lb-C1PVASS into a PDA $\mathcal{P}$ such that the *Parikh image* of a word accepted by the PDA satisfies some quantifier-free Presburger formula φ if and only if the lb-C1PVASS has an accepting run. Then, we use a construction from [18, Theorem 4] (later corrected in [12]) to obtain, in polynomial time, an existential Presburger formula φ_L whose models correspond to the Parikh images of words in the language of $\mathcal{P}$. The problem thus reduces to checking satisfiability of the existential Presburger formula $\varphi \wedge \varphi_L$. The result follows since satisfiability for such formulas is known to be NP-complete [10].

▶ **Theorem 22.** *Deciding k-coverability and k-reachability for lb-C1PVASS is NP-complete, and boundedness is coNP-complete.*

4.1 k-coverability for lb-C1PVASS

For lb-C1PVASS, k-coverability is equivalent to *state reachability*, i.e. without asking for the final counter value to be at least some given value: to check k-coverability, we add a new final state with lower-bound guard k and transitions from the old final state(s) to this new state with $+0$ counter updates and no stack update. Because of this, we focus on state reachability as acceptance condition and omit k when speaking of coverability in the sequel.

Let $\mathcal{A} = (S, \Gamma, \delta, \perp, s_0, f, \ell)$ be the lb-C1PVASS. Recall that $\ell : S \to \mathbb{N}$ is the mapping from states to the lower bounds on those states. That is, $\ell(s) = x$ implies that the counter value must be at least x in order to enter the state s. Let $n = |S|$ be the number of states. We have the assumption, from Lemma 6, that the only counter updates in the lb-C1PVASS are in the set $\{-1, +0, +1\}$. Let $m + 1 \leq n$ be the size of the range of ℓ. That is, there are $m + 1$ distinct lower bounds $0 = \ell_0 < \ell_1 < \ell_2 < \cdots < \ell_m$ that occur in the lb-C1PVASS $\mathcal{A}$. Note that 0 must be one of the lower bounds since $\ell(s_0) = 0$ in order for any run to exist.

Now, we construct the PDA, followed by the Presburger formula. The PDA $\mathcal{P}$ has $4(m + 1)$ "blocks" and its alphabet is $\Sigma = \{a_i, a'_i, b_i \mid 0 \leq i \leq m + 1\}$. Each block is a subPDA (so, we ignore counter updates) of the lb-C1PVASS with some restrictions. For each $0 \leq i \leq m$, the 4 types of blocks we use all have copies of the same set of states: all $s \in S$ such that $\ell(s) \leq \ell_i$.

1. $\mathcal{I}_i$ The transitions come from those in $\mathcal{A}$ with counter update $+0$ and they read ϵ in $\mathcal{P}$;
2. $\mathcal{I}_i^+$ The transitions come from those in $\mathcal{A}$ with $+0$ or $+1$ updates and the PDA $\mathcal{P}$ reads an a_i on the $+1$ transitions;
3. $\mathcal{A}_i^-$ The transitions come from those in $\mathcal{A}$ with $+0$ or -1 updates and the PDA $\mathcal{P}$ reads a b_i on the -1 transitions;
4. $\mathcal{A}_i^\pm$ And here, all transitions in $\mathcal{A}$ are present and the PDA will read b_i on -1 and a_i on the $+1$ transitions.

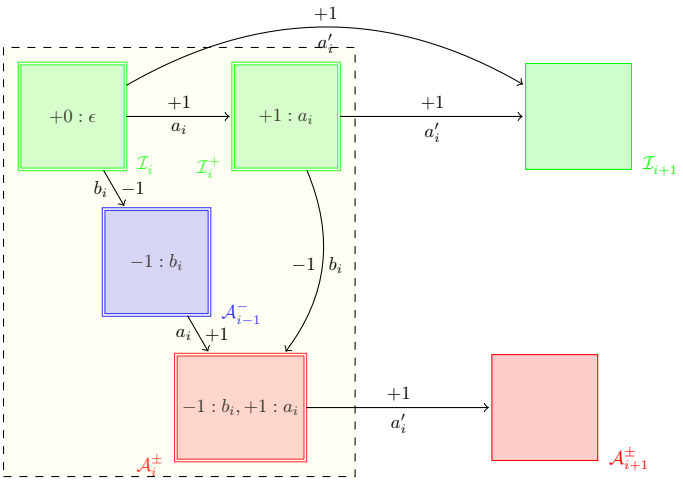

Figure 3 A slice of the PDA $\mathcal{P}$ constructed for k-coverability of a lb-C1PVASS. The subscript being i for $0 \leq i \leq m$ of a block (for example, i in $\mathcal{I}_i$) denotes that all the states in the block have lower bounds at most ℓ_i. Note $+0:\epsilon$ is omitted unless it is the only option for transitions in the block.

Figure 3 depicts how the blocks are connected in what we henceforth call a *slice* (depicted by the dashed box), i.e. $\mathcal{I}_i$, $\mathcal{I}_i^+$, $\mathcal{A}_{i-1}^-$ and $\mathcal{A}_i^\pm$, for some $0 \leq i \leq m$. It also shows how the slices themselves are connected. Note that the transitions in the PDA do not actually have the counter updates $-1, +0, +1$, but we include them in the explanation and figure for clarity. The accepting states of $\mathcal{P}$ are all the copies of accepting state in $\mathcal{A}$.

Now, we define a Presburger formula for the Parikh images of accepting runs in $\mathcal{P}$ that correspond to the accepting runs in $\mathcal{A}$ ($\#_a$ denotes the number of a's read during the run).

$$\bigwedge_{k=1}^{m} \left(\begin{array}{r} (\#_{a'_{k-1}} = 0) \vee \left(\left(\sum_{i=0}^{k-1} \#_{a_i} + \#_{a'_i} \geq \ell_k \right) \wedge \left(\sum_{i=0}^{k-1} \#_{b_i} = 0 \right) \right) \vee \\ \left(\left(\sum_{i=0}^{k-1} \#_{a_i} + \#_{a'_i} > \ell_k \right) \wedge \left(\sum_{i=0}^{k-1} \#_{b_i} > 0 \right) \right) \end{array} \right) \tag{2}$$

Intuitively, the second disjunct ensures that if the run saw no negative updates (stayed in the green layer), then the number of $+1$ updates in order to enter the k^{th} slice must be at least the k^{th} lower bound; The third disjunct ensures that if there was a negative update seen then the number of $+1$ updates seen must be strictly greater than the k^{th} lower bound; and the first disjuct is if the run never enters the k^{th} slice.

▶ **Theorem 23.** *For all runs π in $\mathcal{A}$, there is one in $\mathcal{P}$ with the same sequence of states whose Parikh image satisfies the Presburger formula from Equation* (2) *if and only if π is accepting in $\mathcal{A}$.*

The following auxiliary lemmas are helpful in the intuition for the proof of the theorem.

▶ **Lemma 24.** *Let $0 < \varepsilon < 1$. For any run π in $\mathcal{A}$, there is another run π' with the same sequence of states such that all the $+1$ counter updates in the run are scaled up to 1 and all the -1 updates in the run are scaled down so as to add up to $-\varepsilon$.*

The proof follows from a few simple observations: Since π is a run and π' is a run with the same sequence of states, the stack will behave the same in both runs. For the counter, since we only have lower bounds, and we chose small coefficients for negative updates, all the counter updates in π' are greater than, or equal to the counter updates in π, hence satisfying all lower bounds along the run.

▶ **Lemma 25.** *A run ends in a green state (i.e., a state in $\mathcal{I}_i$ or $\mathcal{I}_i^+$ for some $0 \leq i \leq m$) in $\mathcal{P}$ if and only if there was no b_j read along the run for all $0 \leq j \leq m$.*

This follows from the construction since there is no path from the blue states (states in $\mathcal{A}_i^-$ for $0 \leq i \leq m$) or the red states (states in $\mathcal{A}_i^\pm$ for $0 \leq i \leq m$) to any green state. Intuitively, the counter values are integers when reaching green states and nonintegers when reaching blue or red states.

This gives us NP inclusion for deciding coverability in lb-C1PVASS.

4.2 k-reachability for lb-C1PVASS

Here, we show that k-reachability for $k \in \mathbb{N}$ is also in NP for lb-C1PVASS. Unlike for C1PVASS, k-coverability does not imply k-reachability in lb-C1PVASS: scaling down the vectors along the entire run can lead to some lower bounds being violated. E.g., consider a lb-C1PVASS with 3 states s_0, s_1 and f with $\ell(s_1) = 1$ and a $+1$ update on both $s_0 \to s_1$ and $s_1 \to f$. For any accepting run π, the counter value at s_1 must be 1. This means that even if the second $+1$ update is scaled down, the counter value at f must be strictly greater than 1. For this lb-C1PVASS, 1-coverability holds but 1-reachability does not.

Like in the previous section, we construct a PDA and a Presburger formula such that the PDA accepts a word that satisfies the Presburger formula if and only if the lb-C1PVASS has a k-reaching run, for some given $k \in \mathbb{N}$. However, the construction is more involved since it is not always possible to reach a specific counter value by scaling down all negative counter updates to arbitrarily small numbers. Instead, we first introduce a normal form of scaled runs (i.e. the sequences of coefficients) that guides our construction for a PDA with no block cycles in the same way Lemma 24 guided our construction for reachability.

4.2.1 The Dense Normal Form (DNF)

We show that all k-reaching runs have a normal form which scales the counter updates in the run so that the positive updates are concentrated towards the start of the run and the negative updates towards the end of the run. Formally, let π be a run in the lb-C1PVASS which reaches the counter value $k \in \mathbb{N}$. Let P_π be the sum of all positive updates in π and N_π be the sum of all negative updates. Define $I_P^\pi = \lfloor P_\pi \rfloor$, $F_P^\pi = P_\pi - I_P^\pi$, $I_N^\pi = \lceil N_\pi \rceil$ and $F_N^\pi = N_\pi - I_N^\pi$. Clearly $I_P^\pi + F_P^\pi + F_N^\pi + I_N^\pi = k$ and, moreover, $I_P^\pi + I_N^\pi = k$ and $F_P^\pi = -F_N^\pi$ since k is an integer. To define a normal form and argue all runs can be put in it, we scale consecutive positive updates and consecutive I_N^π negative updates at the start and at the end the run in full, i.e. $\gamma = 1$ (except for a special case where we scale one of the negative updates by Δ close to 1). The remaining positive and negative updates can be scaled arbitrarily (small) as long as their sum adds up to 0. For the latter, we will scale down positive and negative updates by coefficients from E and D respectively, where E and D are finite sets of arbitrarily small "epsilons" (ε) and "deltas" (δ) in the interval $(0, 1)$. The updates are concentrated at the start and the end of the run, hence the name *dense* normal form. For simplicity, we also add Δ to D.

▶ **Lemma 26.** *Let π be a run in the lb-C1PVASS which reaches the counter value $k \in \mathbb{N}$. Then, there exists a run π' such that:*

1. *The sequence of states in π' is the same as in π, and*
2. *π' is also a k-reaching run.*
3. *All positive and negative updates in π' are scaled from the set $\{1\} \cup E$ and $\{1\} \cup D \cup \Delta$, respectively.*
4. *The sequence of all nonzero updates in π' is of the form $(\{+1\} \cup -D)^*(-D \cup +E)^*(\{-1\} \cup +E)^*$ or $(\{+1\})^*(-D)^+(-\Delta)(\{-1\})^*$.*
5. *Let $\pi'|_{E \cup D}$ be the sequence of counter updates restricted to $+E$'s and $-D$'s. Then, $\pi'|_{E \cup D}$ is of the form:*
   - *$(-D)^*(+E)(+E \cup -D)^*(-D)$, in which case, for any proper prefix of $\pi'|_{E \cup D}$ with at least one epsilon, the sum of all epsilons and deltas is positive; or*
   - *$(+E \cup -D)^*(-D)(+E)^+$, and for any proper prefix of $\pi'|_{E \cup D}$ that contains an epsilon and not the final delta, the sum of all epsilons and deltas is positive while for any proper prefix with the final delta, the sum is negative; or*
   - *$(-D)^+(-\Delta)$, and none of the deltas occur before a $+1$ counter update.*
6. *For configurations $q = (s, \alpha, c)$ in π and $q' = (s, \alpha, c')$ in π' which occur in the same position in both runs, $c' \geq \lfloor c \rfloor$.*

Note that the DNF (in particular Item 4) precludes -1 updates before a $+1$ update. It also visits the same sequence of states, satisfies all lower-bound guards along the run and reaches the same counter value k as the given run.

▶ **Example 27.** Let the sequence of nonzero updates in a 1-reaching run be $+1$, $+0.8$, -0.9, -0.9, $+1$, -1, $+1$. This run is not in the dense normal form. To transform it into the normal form, we choose different values of γ to scale the updates along the run to obtain $+1, +1, -\delta_1, -\delta_2, +\varepsilon_1, -1, +\varepsilon_2$. It is easy to see that this is also a 1-reaching run and the integer parts of the counter values along the run are at least those along the original run.

Since, if a k-reaching run exists, a k-reaching run in dense normal form exists, we construct a PDA and a Presburger formula which simulate runs of the lb-C1PVASS in DNF.

4.2.2 Constructing the PDA $\mathcal{P}$

The construction of $\mathcal{P}$ is shown in Figure 4. Each slice consists of 9 layers, namely $L_1, \ldots, L_9$, each of which contains one or two blocks. Each block, like in Section 4.1 is a copy of the lb-C1PVASS $\mathcal{A}$ restricted to states with lower bounds at most ℓ_i, where i is the subscript of the block, and transitions with updates written inside the block. The transitions labelled $+\varepsilon$ and $-\delta$ correspond to $+1$ and -1 update transitions respectively, but we label them differently to make the explanation later easier to read. Each block also has all transitions from $\mathcal{A}$ with a $+0$ update. Note that the connections between blocks allow exactly the transitions on the labels and not the $+0$ transitions from $\mathcal{A}$.

In the figure, we are assuming that $\ell_{i-1} < \ell_i - 1$. If $\ell_{i-1} = \ell_i - 1$, there will be the following changes:

1. All transitions entering $\mathcal{A}^-_{i-1}$ will now go to $\mathcal{A}^-_{i-2}$ instead (both in L_7), and
2. All transitions entering $\mathcal{I}^-_{i-1}$ will now go to $\mathcal{I}_{i-1}$ instead.

Checking whether $\ell_{i-1} = \ell_i - 1$ can be done beforehand for all $0 < i \leq m$, and $\mathcal{P}$ can be constructed accordingly. The discussion that follows still holds.

Due to the complexity of the PDA, a fully formal proof like in Section 4.1 would obscure rather than support our claims. Instead, we focus on providing the high-level intuition behind the construction of the PDA and the Presburger formula.

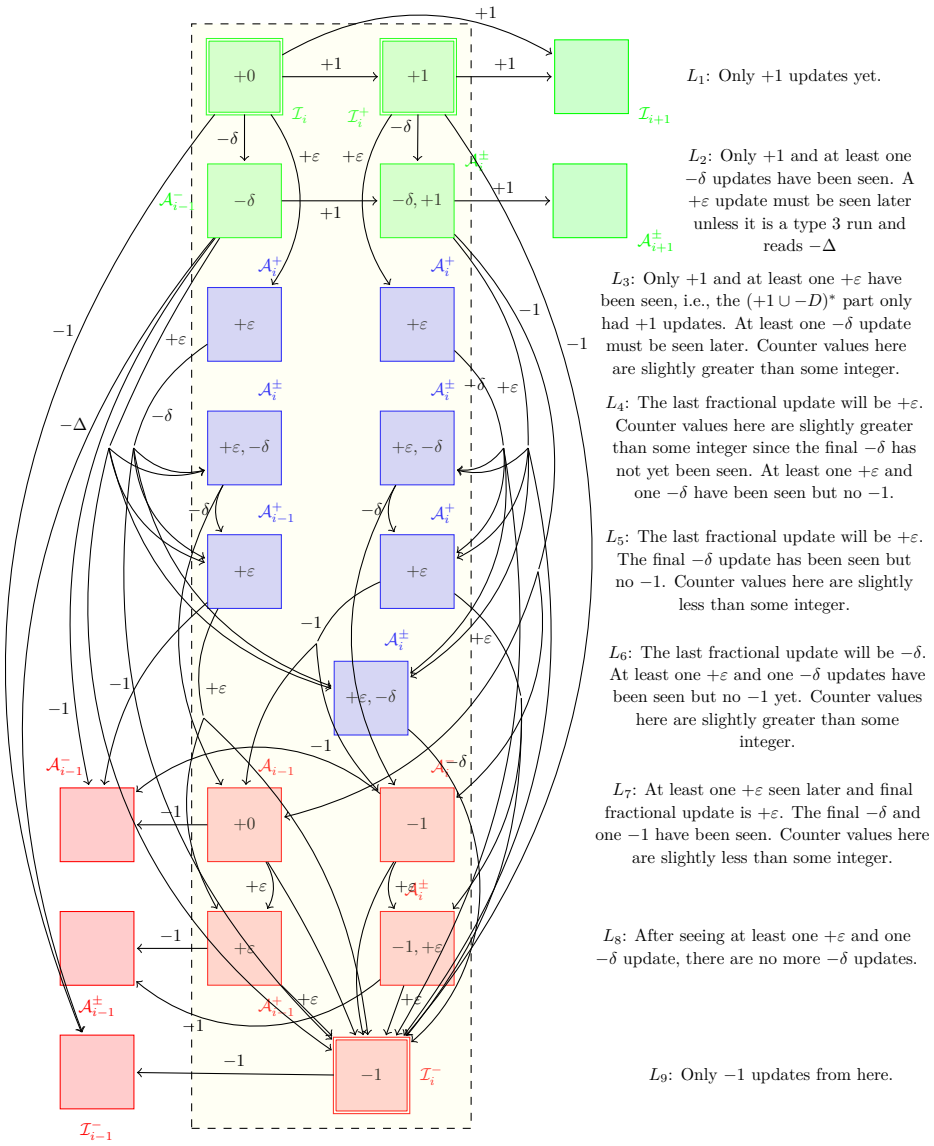

Figure 4 The i^{th} slice of the PDA $\mathcal{P}$ constructed for k-reachability of a lb-C1PVASS. The slice itself is inside the dashed box, the text to the right provides intuition for the layer. The subscript being i for $0 \leq i \leq m$ of a block (eg, i in $\mathcal{I}_i$) denotes that all the states in the block have lower bounds at most ℓ_i. Note $+0$ is omitted unless it is the only option for transitions in the block.

▶ Remark 28. By construction, there are no block cycles in $\mathcal{P}$. That is, once a run exits a block, it cannot enter the same block later. This follows due to the simple observation that, from every block, there are transitions only to a block either to the left or to the right on the same level (but never both), or a block in a lower layer.

Note that, since there is no way to enter a block in an upper layer from a lower one, and a run always starts from $\mathcal{I}_0$ (the leftmost green block), any run will first traverse green blocks, moving to the right, then it will either enter a blue block and move down or directly enter a red block and start moving to the left.

▶ **Lemma 29.** *The sequence of states in any run in $\mathcal{P}$ is a sequence of states in green blocks, followed by a (possibly empty) sequence of states in blue blocks, followed by a (possibly empty) sequence of states in red blocks. Furthermore, while the run is visiting green, blue and red blocks, the index of the slices is non-decreasing, constant and non-increasing, respectively.*

The proof follows by construction.

We now show how the PDA $\mathcal{P}$ is split into 3 main components and the letters read on the transitions in slice i.

- The green component consists of the first two layers. This component corresponds to the $(+1 \cup -D)^*$ part of the run in Item 4. This is easy to see since this component looks exactly like Figure 3, with the exception that the states in the second layer (which were the states in blue and red states in Figure 3), are not accepting. The PDA reads alphabet a_i on all $+1$ transitions in and to green blocks in Figure 4, except the transitions entering the next slice, on which it reads a'_i, and it reads d_i on all the $-\delta$ transitions.
- The blue component consists of layers 3, 4, 5 and 6 which correspond to the $(+E \cup -D)^*$ part of the run. This component is entered after reading the first $+\varepsilon$ update. Due to Item 5, the run stays in a single slice during this part of the run. The empty letter ϵ is read on all transitions in and to this component.
- The red component, consisting of the last 3 layers, corresponds to the $(-1 \cup +E)^*$ part of the run, and is entered after the first -1 or the last fractional update. Note that the run can never exit this component once it is entered. The PDA reads b'_i on all the -1 transitions in and to L_7, b_i on all -1 transitions in and to L_8 and L_9, ϵ on all other transitions and a d'_i on the $-\Delta$ transition entering L_9. Note that $\sum_{i=1}^{m} \#_{d'_i} \in \{0, 1\}$ since a $-\Delta$ transition can occur at most once in any run.

▶ **Lemma 30.** *If the lb-C1PVASS accepts some run in DNF which reaches some $k \in \mathbb{N}$, there exists an accepting run in the PDA $\mathcal{P}$ such that $\sum_{i=1}^{m} \#_{a_i} + \#_{a'_i} - \#_{b_i} - \#_{b'_i} - \#_{d'_i} = k$.*

The proof follows from the fact that the run (in DNF) can be simulated on the PDA $\mathcal{P}$ by staying in the green component for the $(-1 \cup -D)^*$ portion of the run, then going to the $(+E \cup -D)^*$ portion of the run, and finally the run enters the red portion for the $(-1 \cup +E)^*$ part of the run (except for the case where the run is of the form $(\{+1\})^*(-D)^+(-\Delta)(\{-1\})^*$, in which case the run enters the last layer on the $-\Delta$ update). The run moves between layers as the counter values move between different slice bounds.

4.2.3 The Presburger formula

The main intuition for having the Presburger formula is to make sure that the run stays in the correct block. For example, on reading a $+1$ in $\mathcal{I}_i^+$, there is a choice to either stay within the block or move to $\mathcal{I}_{i+1}$. The formula ensures that it stays in $\mathcal{I}_i^+$ when the counter value is in the interval (ℓ_i, ℓ_{i+1}) and moves to $\mathcal{I}_{i+1}$ when the counter value reaches ℓ_{i+1}.

We also use a formula to ensure that the sum of all the $+1$ and -1 updates is exactly k.

Using Lemma 29, we are able to split the Presburger formula into 4 conjuncts as well.

$$\varphi_G = \bigwedge_{k=1}^{m} \left(\begin{array}{c} \left(\#_{a'_{k-1}} = 0\right) \vee \left(\left(\sum_{i=0}^{k-1} \#_{a_i} + \#_{a'_i} \geq \ell_k \right) \wedge \left(\sum_{i=0}^{k-1} \#_{d_i} = 0 \right) \right) \vee \\ \left(\left(\sum_{i=0}^{k-1} \#_{a_i} + \#_{a'_i} > \ell_k \right) \wedge \left(\sum_{i=0}^{k-1} \#_{d_i} > 0 \right) \right) \end{array} \right)$$

$$\varphi'_R = \bigwedge_{k=1}^{m} \left(\left(\#_{b'_k} = 0 \right) \vee \left(\sum_{i=1}^{m} \#_{a_i} + \#_{a'_i} - \sum_{j=k}^{m} \#_{b'_j} > \ell_k \right) \right)$$

$$\varphi_R = \bigwedge_{k=1}^{m} \left(\left(\#_{b_k} + \#_{d'_{k+1}} = 0 \right) \vee \left(\sum_{i=1}^{m} \#_{a_i} + \#_{a'_i} - \sum_{j=k}^{m} \#_{b'_j} + \#_{b_j} + \#_{d'_{j+1}} \geq \ell_k \right) \right)$$

$$\varphi_k = \sum_{i=0}^{m} \#_{a_i} + \#_{a'_i} - \#_{b_i} - \#_{b'_i} - \#_{d'_i} = k$$

The final formula φ will be a conjunction of all the formulas described above.

$$\varphi = \varphi_G \wedge \varphi'_R \wedge \varphi_R \wedge \varphi_k. \tag{3}$$

Now, with the PDA $\mathcal{P}$ and formula φ, we get the main result of this section.

▶ **Theorem 31.** *For all runs π in $\mathcal{A}$, there is one in $\mathcal{P}$ with the same sequence of states whose Parikh image satisfies the Presburger formula φ from Equation (3) if and only if π is k-reaching in $\mathcal{A}$.*

The proof is similar to that of Theorem 23, simulating the k-reaching runs of $\mathcal{A}$ using accepting runs in $\mathcal{P}$ satisfying φ and vice-versa.

4.3 Boundedness for lb-C1PVASS

For boundedness, we first note that the set of all reachable counter values from a lb-C1PVASS $\mathcal{A}$ is a subset of all reachable values of the C1PVASS $\mathcal{A}'$, where $\mathcal{A}'$ is obtained by replacing the lower bounds in $\mathcal{A}$ by 0. It follows, from Theorem 17, that $2^{O(|\mathcal{A}|^6)}$ is an upper bound on the largest reachable counter value in $\mathcal{A}$, if $\mathcal{A}$ is bounded, where $|\mathcal{A}|$ is the size of the encoding of $\mathcal{A}$. Hence, we can ask for k-coverability with $k = 2^{O(|\mathcal{A}|^6)}$ and if the answer is yes, the set of all reachable counter values is unbounded. Hence, boundedness is in coNP.

4.4 Hardness of k-reachability, k-coverability and boundedness

We now show that reachability and coverability are NP-hard for lb-C1PVASS and coNP-hard for boundedness. This gives us NP-completeness for reachability and coverability, and coNP-completeness for boundedness.

▶ **Lemma 32** (Context-free sum problem). *Given a context-free language $\mathcal{L} \subseteq a^*b^*$ and two numbers $k, \ell \in \mathbb{N}$ (encoded in binary), determining whether there exists a word $a^m b^n \in \mathcal{L}$ such that $m \geq k$ and $m + n \geq \ell$ is NP-hard.*

Proof. We reduce from the subset sum problem: Given a set of positive integers $S = \{x_1, \ldots, x_n\}$ and an integer X, is there a subset $S' \subseteq S$ such that $\sum_{x_i \in S'} x_i = X$?

We construct the CFG as follows: We have non-terminals $A_0, \ldots, A_n$ where A_0 is the start symbol, and productions $A_i \to a^{x_{i+1}} A_{i+1}$, $A_i \to A_{i+1} b^{2x_{i+1}}$, for all $0 \leq i \leq n-1$, and $A_n \to \varepsilon$. This CFG defines the following language: $\mathcal{L} = \{a^r b^s \mid \exists S' \subseteq S : r = \sum_{x_i \in S'} x_i \text{ and } s = 2\sum_{j \notin S'} x_j\}$. One can easily verify that a word $a^n b^m$ satisfying $n \geq X$ and $n + m \geq 2\sum_{i=1}^{n} x_i - X$ is in $\mathcal{L}$ if and only if the subset sum instance is positive. ◀

We now give the reduction from this problem to reachability, coverability and unboundedness in an lb-C1PVASS. Without loss of generality, we assume that the context-free language is given as a PDA such that there is a partition between all the states with an outgoing transition labelled by an a and those with a transition labelled by a b as shown in Figure 5.

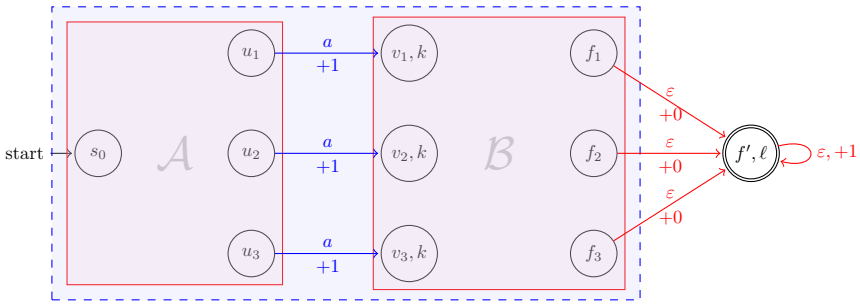

Figure 5 The construction of the lb-C1PVASS where $\mathcal{A}$ is the part of the PDA consisting of only the transitions which do not read b's and on the last a, there is a check verifying that at least k many a's were seen, before seeing the bs. From the final states, there is a transition to the new final state f' with a lower bound of ℓ which checks that the total number of a's and b's is at least ℓ.

To show that this construction establishes a reduction from the context-free sum problem to reachability, coverability and unboundedness, it is enough to see that an accepting run from the PDA will be accepting in this C1PVASS if and only if the total number of a's is at least k and the total number of a's and b's is at least ℓ. Thus ℓ is reachable if and only if it is coverable if and only if there exists a solution to the context-free sum problem. Finally, due to the $+1$ self loop on the final state, there is are accepting runs with unbounded value if and only if the context-free sum instance is positive. This gives us NP-hardness for reachability and coverability, and coNP-hardness for boundedness.

5 Conclusion

In this work we established reachability, coverability, and boundedness are decidable in polynomial time for continuous PVASS in one dimension (C1PVASS). When the model is extended with lower-bound guards for the counter on the states, we proved reachability and coverability are NP-complete while boundedness is coNP-complete. There are cells of Table 1 for which complexity bounds can be tightened in the future. Our algorithms can be used as heuristics to guide the exact algorithm for the reachability problem if it is decidable (cf. [3]). In the direction of using C1PVASS as approximations of PVASS, we posit the most interesting direction is to add both upper and lower bounds to the values the counter can take (cf. [4]) towards an approximation of one-counter pushdown automata. This model with upper and lower bounds can be seen as a generalization of one clock pushdown timed automata without resets.

References

1. A. R. Balasubramanian. Decidability and complexity of decision problems for affine continuous VASS. In Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza, editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 7:1–7:13. ACM, 2024. doi:10.1145/3661814.3662124.
2. A. R. Balasubramanian, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. Reachability in continuous pushdown VASS. *Proc. ACM Program. Lang.*, 8(POPL):90–114, 2024. doi:10.1145/3633279.
3. Michael Blondin, Christoph Haase, and Philip Offtermatt. Directed reachability for infinite-state systems. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part II*, volume 12652 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2021. doi:10.1007/978-3-030-72013-1_1.
4. Michael Blondin, Tim Leys, Filip Mazowiecki, Philip Offtermatt, and Guillermo A. Pérez. Continuous one-counter automata. *ACM Trans. Comput. Log.*, 24(1):3:1–3:31, 2023. doi:10.1145/3558549.
5. Dmitry Chistikov and Rupak Majumdar. Unary pushdown automata and straight-line programs. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming*, pages 146–157, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
6. Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for petri nets is not elementary. *J. ACM*, 68(1):7:1–7:28, 2021. doi:10.1145/3422822.
7. Matthias Englert, Piotr Hofman, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Juliusz Straszyński. A lower bound for the coverability problem in acyclic pushdown vas. *Information Processing Letters*, 167:106079, 2021.
8. Yuval Filmus. Hardness of finding a word of length at most k accepted by a nondeterministic pushdown automaton. https://cstheory.stackexchange.com/questions/4429/hardness-of-finding-a-word-of-length-at-most-k-accepted-by-a-nondeterministic, 2011.
9. Moses Ganardi, Rupak Majumdar, Andreas Pavlogiannis, Lia Schütze, and Georg Zetzsche. Reachability in bidirected pushdown VASS. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 124:1–124:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.124.
10. Christoph Haase. A survival guide to presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018.
11. Matthew Hague and Anthony Widjaja Lin. Model checking recursive programs with numeric data types. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 743–759. Springer, 2011. doi:10.1007/978-3-642-22110-1_60.
12. Matthew Hague and Anthony Widjaja Lin. Synchronisation- and reversal-bounded analysis of multithreaded programs with counters. In *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2012.
13. John E. Hopcroft and Jeff D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, 1979.

14 Jérôme Leroux, M. Praveen, and Grégoire Sutre. Hyper-ackermannian bounds for pushdown vector addition systems. 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 63:1–63:10. ACM, 2014. doi:10.1145/2603088.2603146.

15 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. 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.2019.8785796.

16 Jérôme Leroux, Grégoire Sutre, and Patrick Totzke. On the coverability problem for pushdown vector addition systems in one dimension. In *International Colloquium on Automata, Languages, and Programming*, pages 324–336. Springer, 2015.

17 Sylvain Schmitz. The complexity of reachability in vector addition systems. *ACM SIGLOG News*, 3(1):4–21, 2016. doi:10.1145/2893582.2893585.

18 Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the complexity of equational horn clauses. In *CADE*, volume 3632 of *Lecture Notes in Computer Science*, pages 337–352. Springer, 2005.

A Appendix

A.1 Proof of Lemma 26

Proof. We first construct a run π'' which satisfies $I_P^{\pi'} = I_P^{\pi}$ and $I_N^{\pi'} = I_N^{\pi}$. Notice that none of the lower bounds along the run π can exceed I_P^{π}.

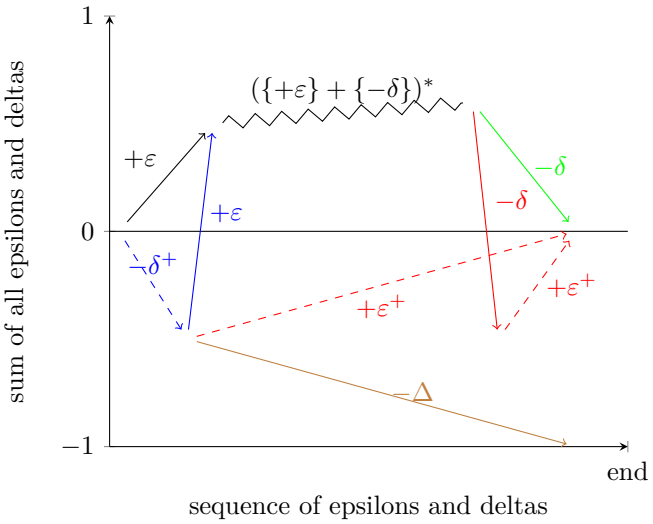

Figure 6 A graphical representation of the multiple cases.

π must have at least I_P^{π} positive and I_N^{π} negative updates due to Lemma 6. The idea is to scale up the first I_P^{π} many $+1$ updates and the last I_N^{π} many -1 updates in the π'' by 1 (call this the *1-scaling*), and scale the remainder of $+1$ and -1 updates by updates in E and D. There are two major cases for this:

1. The I_P^{π}'th 1-scaled $+1$ update occurs before the first 1-scaled -1 update.
2. The I_P^{π}'th 1-scaled $+1$ update occurs after the first 1-scaled -1 update.

If the first case holds, π'' is of the form $(\{+1\} \cup -D)^*(-D \cup +E)^*(\{-1\} \cup +E)^*$ and we scale the updates in E and D according to Figure 6 on a case-by-case basis:

- There are exactly I_P^π many positive updates and I_N^π many negative updates in π (which means that all of them must have been scaled to 1 and $F_P^\pi = F_N^\pi = 0$), $\pi'' = \pi' = \pi$ and the bounds are satisfied.
- The number of positive updates in π is strictly greater than I_P^π and has exactly I_N^π many negative updates. π'' is of the form $(+1)^*(\{-1\} \cup E)^*$. Note that π'' does not reach k, but $k + e$ where e is the sum of all epsilons in the run. Since all the -1 updates are 1-scaled even in π, there must be some update in the first I_P^π $+1$ updates which is not 1-scaled. Also, the lower bound in any state in π is at most $I_P^\pi - 1$. For π', we scale down the I_P^π'th $+1$ update in π'' to $+\varepsilon$ (the solid black edge in Figure 6) and the first 1-scaled -1 update to $-\delta$, such that the sum of all the epsilons and deltas is now 0. The lower bounds until the I_P^π'th positive update (i.e., the first epsilon update) are satisfied since they were satisfied in π and the counter value after this sequence of update will be $I_P^\pi - 1 + \varepsilon$. The counter values until the (only) delta updates will be satisfied since the counter values will be between $I_P^\pi - 1$ and I_P^π. After the delta update, the counter values are satisfied in π' because they were satisfied in π, both runs reach k and π' scales up the -1 updates by 1 and scaled down the $+1$ updates to ε in this section of the run (intuitively, reaching the same number with lower positive updates and higher negative updates implies that the counter values must be better).
- The number of negative updates in π is strictly greater than I_N^π and has exactly I_P^π many positive updates. π'' is of the form $(\{+1\} \cup D)^*(-1)^*$. Note that π'' does not reach k, but $k - d$ where d is the sum of all deltas in the run. Here, we encounter the special case which calls for the need of Δ, that is there might be no negative update in π before a positive update, in which case, the I_P^π'th $+1$ update may not be able to be scaled down in π' since the lower bound after the I_P^π'th $+1$ update might be I_P^π. If this is the case, π' 1-scales all of the $+1$ updates and the last $I_N^\pi - 1$ many -1 updates. The remaining -1 updates are scaled to some small values in D (the dotted blue edge in Figure 6) except the last one, which is scaled to Δ (the brown edge in Figure 6). Thus, the sum is still k since all of the negative updates which are not 1-scaled add up to exactly -1. The lower bounds in this case are satisfied since they were satisfied in π and all of the $+1$ updates are scaled to 1 and the negative updates are postponed as much as possible.

  If this is not the case, that is the lower bound after the I_P^π'th positive update is not I_P^π, it follows that none of the lower bounds along π are I_P^π. In π', we 1-scale the first I_P^π many $+1$ updates and the last $I_N^\pi - 1$ many -1 updates, scale down the I_P^π'th $+1$ update to a value in E (the solid black edge or the solid blue edge in Figure 6) in π', and scale the -1 updates which were not 1-scaled to small values in D (the zigzag line and the green edge in Figure 6). The lower bounds are satisfied until the (only) $+\varepsilon$ since they were satisfied in π, all $+1$ updates are scaled to 1 and all negative updates are scaled down to small values in D. The lower bound immediately after the $+\varepsilon$ update is satisfied since the counter value at this point is between $I_P^\pi - 1$ and I_P^π. The lower bounds in the remainder are satisfied since π and π' both reach k, there are no positive updates and π' postpones the negative updates as much as possible.

  For the remaining cases, we assume the number of positive and negative updates in π is strictly greater than I_P^π and I_N^π respectively. We also fix $\pi' = \pi''$ for these cases and hence omit the use of π''.
- After the 1-scaling, the first non-zero update remaining is a $+1$ and the last one is a -1 (i.e., the first scaled down update in π' is a $+\varepsilon$ and the last one is a $-\delta$). Since the first nonzero update in the run apart from the 1-scaled ones is a $+1$ and this update must occur after all the 1-scaled $+1$ updates (because we scale the first I_P^π consecutive updates), this implies the absence of a negative update until the I_P^π'th $+1$ update. The bounds in π' until this point are satisfied because they were satisfied in π, π' scales all the

$+1$ updates to 1 and the remaining updates are $+0$. The counter value at this point is I_P^π. The first $+1$ update after this sequence is scaled down to some $\varepsilon \in E$ (denoted by the first black edge in Figure 6). Similarly, since the last nonzero update apart from the 1-scaled ones is a 11 and this update must occur before all the 1-scaled $+1$ updates (because we scale the last I_N^π consecutive updates), this implies the absence of a negative update after the I_N^π'th -1 update from the end. The last -1 update after this sequence is scaled down to some $\delta \in D$ (denoted by the green edge in Figure 6). The lower bounds until the first -1 update which was 1-scaled are satisfied since the counter values always stayed above I_P^π. The lower bounds in the final section of the string of -1 1-scaled updates are also satisfied since they were satisfied in π, π' delayed the -1 updates as much as possible and they both reach the same value k.

- After the 1-scaling, the first non-zero update remaining is a -1 and the last one is a $+1$ (i.e., the first scaled down update in π' is a $+\delta$ and the last one is a $-\varepsilon$). The lower bounds in the first section with the I_P^π many 1-scaled $+1$ updates (and possibly some $-\delta$ updates) are satisfied because they were satisfied in π and π' scales the $+1$ updates to 1 and the -1 updates to $-\delta$ whose sum never crosses -1 (as depicted by the dashed blue lines in Figure 6). The bounds are satisfied up to the first $+\varepsilon$ update by the same argument, even if there are more $-\delta$ updates after the I_P^π'th 1-scaled $+1$ update. Similarly, the bounds after the last $-\delta$ update are satisfied since they are satisfied in π, both π and π' reach k, and all -1 updates are scaled up by 1 and all $+1$ updates are scaled down to $+\varepsilon$. Now, we only need to show that the bounds were satisfied from the first $+\varepsilon$ to the final $-\delta$. If there was exactly one $+\varepsilon$ (the solid red edge labelled ε in Figure 6), then the bounds are satisfied since it must occur after the last $-\delta$ update. Else, the first $+\varepsilon$ (the solid blue edge in Figure 6) update takes the counter value over I_P^π, it stays there until the final $-\delta$ and the counter values are satisfied.
- For the remaining 2 cases, namely, the first and last scaled down updates in π' are both in $+E$ or $-D$, the proof follows from a permutation of the arguments in the previous two cases.

Now that we have shown that the bounds are satisfied in π', one can easily check that the rest of the properties are satisfied by π' by construction.

Next, if the second case holds, that is there exists a 1-scaled -1 update before a 1-scaled $+1$ update, π'' does not satisfy Item 4 of the normal form. The run π'' is of the form $(\{+1\} \cup -D)^* \cdot -1 \cdot \{+1, -1\}^* \cdot +1 \cdot (\{-1\} \cup +E)^*$. π'' does not necessarily satisfy the lower bounds, nor reach k, but it does have the property that $I_P^{\pi''} - I_N^{\pi''} = k$. We construct π' using π'' and show that the lower bounds hold sequentially. First, the bounds along the $(\{+1\} \cup -D)^*$ section of π'' are satisfied since they were satisfied in π, all $+1$ updates are 1-scaled and all -1 updates are scaled down to values in D. Similarly, the bounds in the $(\{-1\} \cup +E)^*$ of π'' are satisfied. These sections are the same for π'. We scale down the last 1-scaled $+1$ and the first 1-scaled -1 update in π'' to a $+\varepsilon$ and $-\delta$ respectively. Thus, these two updates are "absorbed" into the $(\{-1\} \cup +E)^*$ and $(\{+1\} \cup -D)^*$ sections of the run respectively. Call this new run π_1''. This run also has the property that the $+1$ updates are concentrated on the left, the -1 updates are concentrated on the right, and $I_P^{\pi_1''} - I_N^{\pi_1''} = k$ (since $I_P^{\pi_1''} = I_P^{\pi''} - 1$ and $I_N^{\pi_1''} = I_N^{\pi''} - 1$). If π_1'' is of the form $(\{+1\} \cup -D)^*(-D \cup +E)^*(\{-1\} \cup +E)^*$, construct a k-reaching run in DNF as in case 1, but with π_1'' as input instead of π. Otherwise, π_1'' is also of the form $(\{+1\} \cup -D)^* \cdot -1 \cdot \{+1, -1\}^* \cdot +1 \cdot (\{-1\} \cup +E)^*$, but the size of the $\{+1, -1\}^*$ section of the run will be strictly smaller than that of π'', and we make π_2'' by absorbing the first 1-scaled -1 and the last 1-scaled $+1$ update. Continue this until a run where the first 1-scaled -1 update occurs after the last 1-scaled $+1$ update. ◀